Desarrollo de un Asistente Virtual turístico para la ciudad de Madrid

Trabajo de Fin de Grado
Curso 2017–2018

Autor
Rocío Santos Buitrago

Director
Pablo Gervás Gómez-Navarro

Grado en Ingeniería Informática
Facultad de Informática
Universidad Complutense de Madrid

Desarrollo de un Asistente Virtual turístico para la ciudad de Madrid

Trabajo de Fin de Grado en Ingeniería Informática
Departamento de Ingeniería del Software e Inteligencia Artificial

Autor
Rocío Santos Buitrago

Director
Pablo Gervás Gómez-Navarro

Convocatoria: *Junio 2018*
Calificación:

Grado en Ingeniería Informática
Facultad de Informática
Universidad Complutense de Madrid

8 de junio de 2018

Autorización de difusión

El abajo firmante, matriculado en el Grado en Ingeniería en Informática de la Facultad de Informática, autoriza a la Universidad Complutense de Madrid (UCM) a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a su autor el presente Trabajo Fin de Grado: "Desarrollo de un Asistente Virtual turístico para la ciudad de Madrid", realizado durante el curso académico 2017/2018 bajo la dirección de D. Pablo Gervás Gómez-Navarro en el Departamento de Ingeniería del Software e Inteligencia Artificial, y a la Biblioteca de la UCM a depositarlo en el Archivo Institucional E-Prints Complutense con el objeto de incrementar la difusión, uso e impacto del trabajo en Internet y garantizar su preservación y acceso a largo plazo.

Rocío Santos Buitrago

8 de junio de 2018

Dedicatoria

A mis padres por ser el pilar fundamental en todo lo que soy. En mi educación, tanto académica como de la vida. Por su incondicional apoyo y confianza perfectamente mantenida a través del tiempo.

A mis abuelos, por quererme y apoyarme siempre.

A mis hermanos, Beatriz, Marta, Patricia y Gustavo, por estar conmigo y escucharme siempre, los quiero mucho.

Todo este trabajo ha sido posible gracias a ellos.

Agradecimientos

Después de un intenso período de nueve meses, hoy es el día: escribo este apartado de agradecimientos para finalizar mi trabajo de fin de grado. Ha sido un gran período de aprendizaje, no solo en el campo científico, sino también a nivel personal. Escribir este trabajo ha tenido un gran impacto en mi y es por eso que me gustaría agradecer a todas aquellas personas que me han ayudado y apoyado durante este proceso.

Me gustaría agradecérselo especialmente a mi tutor, Pablo Gervás. Agradecerte tu cooperación y darte las gracias por todas las oportunidades que me has dado durante la investigación. Definitivamente, me brindaste todas las herramientas necesarias para completar mi trabajo de fin de grado satisfactoriamente.

También quiero mostrar mi gratitud a Narciso Martí Oliet por sus muchos y buenos consejos.

También me gustaría agradecer a mis padres sus consejos y su comprensión. Siempre habéis estado ahí para mí.

¡Muchas gracias a todos!

Resumen

Un asistente virtual o *chatbot* es un agente que permite al usuario mantener una conversación "inteligente" utilizando lenguaje natural, hablado o escrito, a través de una aplicación que puede integrarse con un determinado canal.

El objetivo de este proyecto es investigar las plataformas que permiten la creación de un asistente virtual y las distintas tecnologías que intervienen en su desarrollo. El resultado final es la implementación del prototipo de un recomendador de museos para la comunidad y ciudad de Madrid.

Se diseña un chatbot integrado en un canal de mensajería para facilitar la comunicación con el usuario. El desarrollo está basado en el servicio *Watson Conversation* de IBM. El asistente recibe la información obtenida a través del procesamiento del lenguaje natural y además gestiona la comunicación entre el usuario y el bot.

Palabras clave

Chatbot, Asistente virtual, Procesamiento de lenguajes naturales, Interacción persona-computadora, Aprendizaje automático, Inteligencia artificial, Watson, IBM Bluemix, Búsqueda de respuestas

Abstract

A virtual assistant or *chatbot* is an agent that allows the user to keep an "intelligent" conversation using natural language, spoken or written, through an application that can be integrated with a certain channel.

The aim of this project is to research different platforms that allow the creation of a virtual assistant and the different technologies that are involved in its development. The final result is the implementation of the prototype of a museum recommender for the community and city of Madrid.

We have designed a chatbot integrated into a messaging channel to facilitate communication with the user. The development is based on IBM's *Watson Conversation* service. The assistant receives the information obtained through natural language processing and also manages the communication between the user and the bot.

Keywords

Chatbot, Virtual assistant, Natural-language processing, Human-computer interfaces, Machine Learning, Artificial Intelligence, Watson, IBM Bluemix, Question-answering

Índice

Índice de figuras

Índice de tablas

Índice de listados

Chapter 1

Introduction

This chapter begins with a brief introduction of the bots to provide an overview of the different types. Then, a section is dedicated to chatbots. Finally, we describe different scenarios where it is possible to find them.

1.1. Bot and types

In the decade of the Seventies the term "bot" was registered for the first time. Bot use has grown rapidly in recent years. Wikipedia provides the definition of Dunham and Melnick: "A bot (apheresis of a robot) is a computer program that automatically performs repetitive tasks via Internet, which would be impossible or very tedious for a person to perform" (Wikipedia, 2013; Dunham y Melnick, 2008).

In other words, bots are computer applications that perform automated tasks, which allow users to interact with them through the applications. The idea is that through a single application you can get everything you want, without having to waste time with search engines.

Not only are there conversational bots, like the chatbots (Salandra, 2017). Later in this chapter, a categorization of the different types of bots is presented.

Although bots do not have a specific limit, they do not necessarily have to use a cognitive service or Artificial Intelligence. Furthermore, they are not based solely on natural language processing, that is, it is not necessary to understand the user since simple searches can be carried out. Not only are they referred to text interfaces, such as text conversations, but they can contain images, audios, etc.

We can distinguish different types of bots according to their characteristics:

CRAWLERS: They collect information from other websites and index it in their database. There are hundreds of web crawlers and bots that

roam the Internet. The most famous one is the *GoogleBot*, also known as "spider", which is a software designed by Google to index new or updated content on the Internet (Wikipedia, 2017).

INFORMATIVE BOTS: They help to manage the information in the channels. For example, Twitter has an informative bot that publishes tweets on a regular basis (Twitter, 2018). But they can also be seen in many other sectors, where they publish posts with their latest news. Another example in the pharmaceutical field is *Bot PLUS App*, which provides information for pharmacists and healthcare professionals about authorized medicines (Consejo General de Colegios Oficiales de Farmacéuticos, 2018).

CHAT BOTS: They are characterized by simulating a conversation with a human. We will dedicate the following section to explain more in detail this type of bot.

TRANSACTIONAL BOTS: They act as intermediaries between people and external media, performing transactional operations in a satisfactory manner. They are part of the area known as automation of business processes. Quite a few companies are starting to offer it and, according to a Gartner study, it is expected that an 85% of transactions will be made through bots by 2020.

HACKER BOTS: Its main function is to distribute viruses and perform fraudulent actions. The bots on infected computers connect to other infected computers to perform attacks autonomously. For example, *botnets* are used in general to make money through deceptive uses (Kaspersky Lab, 2013). Among the most common uses are mining and cryptocurrency theft and distributed denial of service (DDoS) attacks.

SPAM BOTS: All unwanted advertising and junk content on the Internet is called *spam*. This spam is generated by *spambots* and *spammers*, which seek illicitly gain (CCM, 2013). They are responsible for browsing the web in the search of email addresses, to add them to lists and then send them to those addresses in bulk.

GAME BOTS: This type of bot allows the user to play and offers a great variety of categories (Aristimuño, 2017).

SCRAPER BOTS: They are usually built to steal personal information such as emails, images, etc. One method of this type of bots is *web scraping*, which is used to extract information from websites (Wikipedia, 2018b). That information from the web is indexed with a robot that can be analyzed and stored in a central database. It is used in most of the search engines.

1.2. Chatbot

"Chatbots" are computer programs that allow to keep a conversation with the user and makes them feel as if they were talking to a human being. Chatbots talk with humans, answer their questions, provide suggestions, facilitate user tasks, help improve customer service, increase reservations and/or sales in businesses, ...

Each program has a "specific personality", that is, internally each one has an associated business logic and can execute the actions as they wish.

Eliza has been considered the first chatbot in history (Norvig, 1992). In comparison with the current chatbots, the limitations of the passage of time can be observed. It worked by maintaining a conversation with the user, recognizing key words and asking about them pretending to be a psychologist. Figure 1.1 shows a chatbot based on Eliza.

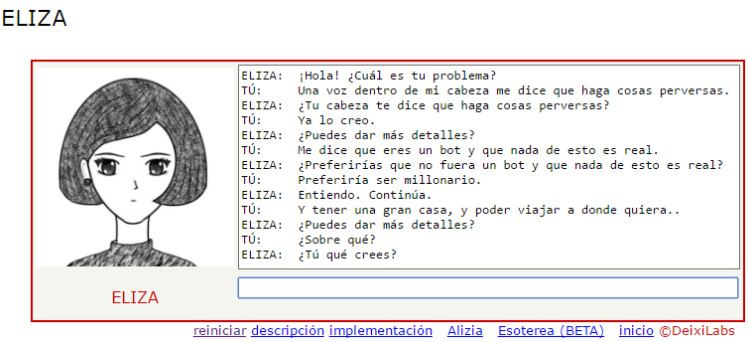


Figure 1.1: Chatbot based on Eliza [`http://deixilabs.com/eliza.html`].

Currently users can interact with chatbots through applications. Normally the applications always perform the same interactions with the users. However, chatbots provide the user with a full control experience (Wikipedia, 2018a). The most advanced chatbots can connect with the user, know their mood and react to those needs; although the bot itself is not able to generate questions or create feelings as a human. Chatbots are computer programs, to which the answers to certain questions are indicated or it answers according to similar words. Most bots fail to fully understand the phrases, or make decisions for themselves.

In terms of learning ability, a chatbot can expand its database with new questions and the use other techniques. One of these techniques is the connection to external services (e.g., cognitive services that allow "bot" to be provided with "intelligence"). An example of applications that use these systems are Siri and Cortana. Figure 1.2 shows some images of chatbots in which you can observe the use of information in real time.

A second case is the chatbots that are used by applications that allow a

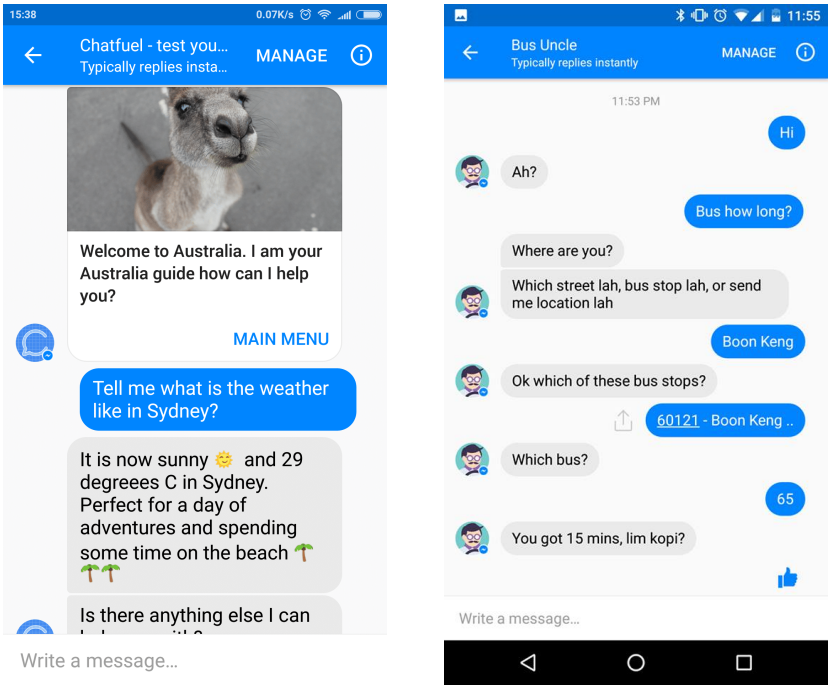


Figure 1.2: Examples of chatbots.

multi-channel broadcast. You can have a chatbot connected to any messaging service, such as Skype, Facebook Messenger, Telegram or Slack, among others (Schlicht, 2016). In these applications, the chatbot is added as a contact. You can also connect to social service APIs such as Facebook or Twitter.

The following chapter will detail examples of the most prominent current chatbots.

1.3. Most common scenarios

In today's society it is usual for many people to spend a large part of the day interacting with their phones. There are numerous activities that users perform frequently and numerous actions that are automated that are periodic and often performed.

A chatbot is a proactive experience that brings new comfort to the user. The applications usually always perform the same interactions and do not give freedom to the user. Other times, we lose a lot of time navigating from one page to another until we find the information we want.

An assistant can help with many tasks, such as providing reminders related to times, places or people. Alarms can be activated with a simple message. It can also track and monitor equipment, interests and flights;

send emails and text messages; keep the user up to date; chat and play; open any application of the system; request reservations at restaurants and hotels; perform marketing activities; etc.

Chatbots can be found in a wide variety of areas, for example, in order fulfillment: a bot is responsible for receiving the order, answering the customer and processing it.

In the field of customer service, a bot can store the frequently asked questions and the necessary information to answer depending on the question. This avoids investing in people with knowledge of the business who answer the users and who always answer the same questions. It also prevents the user from having to be on the phone for hours before being attended.

In the requests of medical assistance, bots facilitate the appointments, they can have an agenda and know when their doctor will be available, etc.

Because of all these facilities of the bots, a boom in their use is expected in the coming years. Users prefer them in searches and access to digital content and services. Besides bots can dispense a personalized treatment that is improved as the bot learns with experience and with the knowledge of the preferences of the user.

1.4. Objectives

The goal of this project is to research the platforms that allow the implementation of a chatbot and the different techniques that are used for its development. In order to show the application of the study, a prototype of a virtual assistant offering tourism services in the Autonomous Community and city of Madrid was developed using existing technologies.

When a tourist is traveling in a new city, questions arise that he needs to solve immediately. Through a simple conversation, the user could obtain both the information about his destination, as the history of the monuments or the characters related to them. The city of Madrid has a large number of activities, for example, a large number of museums to visit and a wide and varied gastronomic offer to eat. In addition, our city has a large number of cinemas, libraries and other tourist services.

Thanks to this chatbot the user can get the recommendation of a museum, based on some attributes, through a simple conversation.

Capítulo 1

Introducción

Este capítulo comienza con una breve introducción a los bots para aportar una visión de los distintos tipos. A continuación se dedica una sección a los chatbots. Por último se describen distintos escenarios donde es posible encontrarlos.

1.1. Bot y tipos

En la década de los setenta se registró por primera vez el término "bot". En la actualidad su utilización es creciente. Wikipedia aporta la definición de Dunham y Melnick: "Un bot (aféresis de robot) es un programa informático que efectúa automáticamente tareas repetitivas a través de Internet, cuya realización por parte de una persona sería imposible o muy tediosa" (Wikipedia, 2013; Dunham y Melnick, 2008).

En otros términos, los bots son aplicaciones informáticas que realizan tareas automatizadas, que permiten a los usuarios interactuar con ellos a través de las aplicaciones. La idea es que a través de una única aplicación se puede conseguir todo lo que se quiere, sin necesidad de malgastar tiempo con buscadores.

No solo hay bots conversacionales, como los chatbots (Salandra, 2017). Más adelante en este capítulo se presenta una categorización de los distintos tipos de bots.

Aunque los bots no tienen un límite concreto, no necesariamente deben utilizar un servicio cognitivo o Inteligencia Artificial. Además no se basan solamente en el procesamiento del lenguaje natural, es decir, no es necesario comprender al usuario ya que se pueden realizar simples búsquedas. No solo están referidas a interfaces de texto, como conversaciones de texto, sino que pueden contener imágenes, audios, etc.

Podemos distinguir distintos tipos de bots en función de sus características:

CRAWLERS: Recogen información de otras webs para indexarla en su base de datos. Hay cientos de rastreadores web y bots que recorren Internet. El más famoso es el *GoogleBot*, también se le conoce como "araña", que es un software diseñado por Google para indexar el contenido nuevo o actualizado de Internet (Wikipedia, 2017).

BOTS INFORMATIVOS: Ayudan a gestionar la información en los canales. Por ejemplo, Twitter dispone de un bot informativo que publica tweets de forma periódica (Twitter, 2018). Pero también se pueden ver en otros muchos sectores, en los que se publican posts con sus últimas novedades. Otro ejemplo en el ámbito farmacéutico es *Bot PLUS App*, que ofrece información para los farmacéuticos y profesionales sanitarios sobre los medicamentos autorizados (Consejo General de Colegios Oficiales de Farmacéuticos, 2018).

CHAT BOTS: Se caracterizan por simular una conversación con un humano. Dedicaremos el siguiente apartado para hablar sobre este tipo con más detalle.

BOTS TRANSACCIONALES: Actúan como intermediarios entre las personas y medios externos, realizando operaciones transaccionales de manera satisfactoria. Forman parte del área conocida como automatización de los procesos de negocio. Un gran número de compañías lo empiezan a ofrecer y, según un estudio de Gartner, se espera que para el año 2020 el 85 % de las transacciones se harán a través de bots.

HACKER BOTS: Su función principal es distribuir virus y realizar acciones fraudulentas. Los bots en equipos infectados se conectan a otros equipos infectados para realizar ataques de forma autónoma. Por ejemplo, los *botnets* son usados en general para ganar dinero a través de usos engañosos (Kaspersky Lab, 2013). Entre los usos más comunes están la minería y robo de criptomoneda y ataques de denegación de servicio distribuidos (DDoS).

SPAM BOTS: A toda la publicidad no deseada y al contenido basura que hay en Internet se denomina *spam* (en castellano, correo no deseado). Este spam es generado por *spambots* y *spammers*, que buscan obtener beneficio de manera ilícita (CCM, 2013). Se encargan de recorrer la web en busca de direcciones de correo electrónico, para agregarlas a listas y luego realizar envíos a esas direcciones de manera masiva.

GAME BOTS: Este tipo de bots permite jugar a sus usuarios y ofrece una gran variedad de categorías (Aristimuño, 2017).

SCRAPER BOTS: Normalmente se construyen para robar información personal como correos, imágenes, etc. Un método de este tipo de bots

es el *web scraping*, que se usa para para extraer información de sitios web (Wikipedia, 2018b). Esa información de la web se indexa con un robot que pueden ser analizarla y almacenarla en una base de datos central. Se utiliza en la mayor parte de los motores de búsqueda.

1.2. Chatbot

Los "chatbots" son programas de ordenador que permiten mantener una conversación con el usuario y le hace sentir como si hablase con un ser humano. Los chatbots conversan con los humanos, dan respuesta a sus dudas, aportan sugerencias, facilitan las tareas del usuario, ayudan a mejorar el servicio al cliente, incrementan las reservas y/o ventas en los negocios, ...

Cada programa tiene una "personalidad específica", es decir, internamente cada uno tiene asociada una lógica empresarial y puede ejecutar las acciones como quiera.

Eliza ha sido considerado el primer chatbot de la historia (Norvig, 1992). En comparación con los chatbot actuales, se pueden observar las limitaciones del paso del tiempo. Funcionaba manteniendo una conversación con el usuario, reconocía palabras claves y preguntaba sobre ellas simulando ser un psicólogo. La Figura 1.1 muestra un chatbot basado en Eliza.

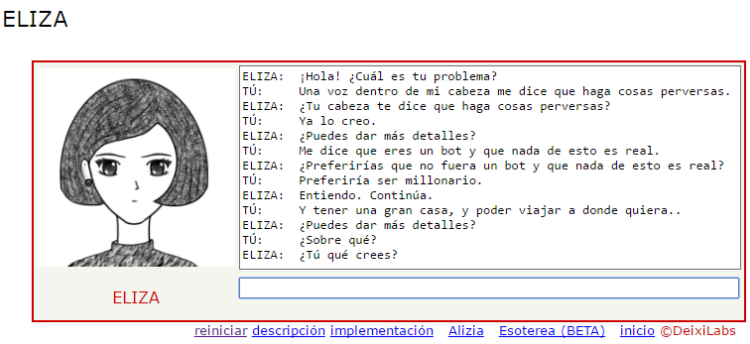


Figura 1.1: Chatbot basado en Eliza [`http://deixilabs.com/eliza.html`].

En la actualidad los usuarios pueden interactuar con los chatbots a través de las aplicaciones. Normalmente las aplicaciones realizan siempre las mismas interacciones con los usuarios. Sin embargo, los chatbots aportan al usuario una experiencia de pleno control (Wikipedia, 2018a). Los chatbots más avanzados pueden conectar con el usuario, conocer su estado de ánimo y reaccionar ante esas necesidades; aunque por él mismo no es capaz de generar preguntas o crear sentimientos como un humano. No deja de ser un ente programado, al que se le indican las respuestas a ciertas preguntas o contesta en función de palabras similares. La mayoría de los bots no consiguen comprender del todo las frases, ni tomar decisiones por sí mismos.

En cuanto a la capacidad de aprendizaje, un chatbot puede ir ampliando su base de datos con nuevas preguntas y utilizar otras técnicas. Una de esas técnicas es la conexión a servicios externos (e.g., servicios cognitivos que permiten dotar de "inteligencia" al bot). Un ejemplo de aplicación que utiliza estos sistemas son Siri o Cortana. La Figura 1.2 muestra unas imágenes de chatbots en las que se puede observar la utilización de información en tiempo real.

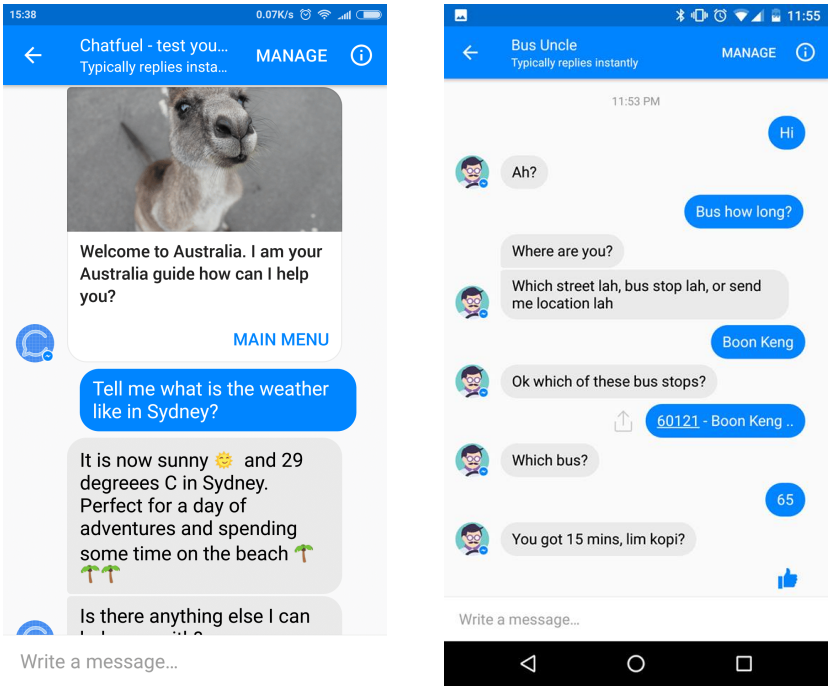


Figura 1.2: Ejemplos de chatbots.

Un segundo caso lo constituyen los chatbots que se utilizan mediante aplicaciones que permiten una difusión multi-canal. Se puede tener un chatbot conectado a cualquier servicio de mensajería, como Skype, Facebook Messenger, Telegram o Slack, entre otras (Schlicht, 2016). En estas aplicaciones se incorpora el chatbot como un contacto más. Además puede conectarse a APIs de redes sociales de servicios como Facebook o Twitter.

En el capítulo siguiente se detallarán ejemplos de los chatbots actuales más destacados.

1.3. Escenarios más habituales

En la sociedad actual para muchas personas es habitual pasar una gran parte del día interaccionando con sus teléfonos. Hay numerosas actividades que los usuarios realizan con frecuencia y numerosas acciones que están au-

tomatizadas que son periódicas y se realizan a menudo.

Un chatbot es una experiencia proactiva que aporta una comodidad novedosa al usuario. Las aplicaciones normalmente realizan siempre las mismas interacciones y no dan libertad al usuario. Otras veces perdemos mucho tiempo con la navegación de una página a otra hasta encontrar la información que se desea.

Un asistente puede ayudar en muchas tareas, como la de facilitar recordatorios relacionados con momentos, lugares o personas. Con un simple mensaje se pueden activar alarmas. También puede realizar el seguimiento y monitorización de equipos, intereses y vuelos; enviar correos electrónicos y mensajes de texto; mantener al usuario al día; charlar y jugar; abrir cualquier aplicación del sistema; solicitar reservas en restaurantes y hoteles; realizar actividades de marketing; etc.

Se pueden encontrar chatbots en una gran variedad de ámbitos, por ejemplo, en el realización de pedidos: un bot se encarga de recibir el pedido, contestar al cliente y tramitarlo.

En el ámbito de la atención al cliente, un bot puede almacenar las preguntas frecuentes y la información necesaria para contestar en función de la pregunta. De esta forma se evita invertir en personas con conocimiento del negocio que contestan a los usuarios y que siempre contestan las mismas preguntas. También se evita que el usuario tenga que estar horas al teléfono hasta ser atendido.

En las solicitudes de asistencia médica, los bots facilitan la reserva de consultas, pueden disponer de una agenda y saber cuándo su médico estará disponible, etc.

Por todas estas facilidades de los bots, se espera un auge de su uso en los próximos años. Los usuarios los prefieren en las búsquedas y en el acceso a los contenidos y servicios digitales. Además los bots pueden dispensar un trato personalizado que se mejora según el bot aprende con la experiencia adquirida y con el conocimiento de las preferencias del usuario.

1.4. Objetivos

El objetivo de este proyecto es investigar las plataformas que permiten la implementación de un chatbot y las distintas técnicas que se utilizan para su desarrollo. Para demostrar la aplicación del estudio se elaborará, mediante las tecnologías existentes, un prototipo de un asistente virtual que ofrezca servicios turísticos de la comunidad y ciudad de Madrid.

Cuando un turista se encuentra en una ciudad, le surgen dudas que necesita resolver al momento. A través de una simple conversación, se podría obtener tanto la información de su destino, como la historia de los monumentos o los personajes relacionados con ellos. La ciudad de Madrid dispone de una gran cantidad de actividades, por ejemplo, numerosos museos y una

amplia y variada oferta gastronómica. Además, nuestra ciudad cuenta con gran número de cines, bibliotecas y otros servicios turísticos.

Gracias a este chatbot se puede obtener la recomendación de un museo, en base a unos atributos, a través de una simple conversación.

Capítulo 2

Estado de la cuestión

La primera sección de este capítulo muestra la evolución de los chatbots desde sus inicios hasta la actualidad. A continuación se analiza cómo los chatbots han llegado a introducirse hoy en día. Se mostrarán también los chatbots más importantes y utilizados en la actualidad.

Por último, se estudiarán las distintas plataformas existentes para la creación de chatbots y las técnicas que se utilizan para desarrollar el diálogo. También se comentan las ventajas y limitaciones que se han encontrado en el desarrollo de estos trabajos.

2.1. Evolución histórica

Para situarnos en el momento actual hay que hacer un recorrido desde principios de los años cincuenta. En 1950 Alan Turing propuso en su ensayo *Computer Machinery and Intelligence* un test para calificar la inteligencia de las máquinas (Turing, 1950). La *prueba de Turing* es uno de los temas más controvertidos en Inteligencia Artificial, filosofía de la mente y ciencia cognitiva (Saygin et al., 2000). Si una máquina fuera capaz de hacernos creer que estamos dialogando con un humano, entonces pasará esta prueba y la consideraremos inteligente (ver Figura 2.1).

Unos años más tarde John McCarthy acuñó el término de Inteligencia Artificial. La describe como la ciencia y la ingeniería de hacer máquinas inteligentes (McCarthy, 1960; Cerdas Mendez, 2016).

Inbenta sitúa la creación de los primeros chatbots junto los principales hitos necesarios que intervienen en el funcionamiento de los chatbots: primer motor de búsqueda, primer *webclawler*, búsquedas con lenguaje natural, motores de búsqueda semánticos, reconocimiento de voz, Inteligencia Artificial del chatbot (Inbenta, 2017). Los primeros chatbots que investigaban el procesamiento del lenguaje natural surgieron a principios de los sesenta (Mauldin, 1994). La Figura 2.2 muestra la cronología de la historia de los chatbots.

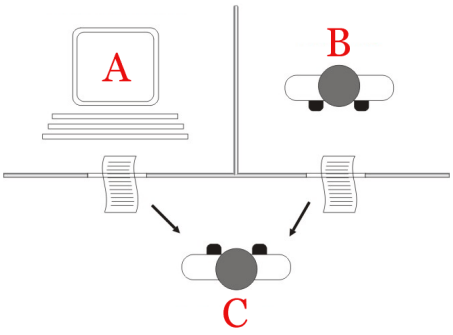


Figura 2.1: *Test de Turing*. La "interpretación estándar" de la prueba de Turing, en la cual el jugador C, el interrogador, le es dada la tarea de tratar de determinar qué jugador —¿A ó B?— es una computadora y cual un ser humano. El interrogador se limita a la utilización de las respuestas a las preguntas escritas para tomar la determinación [Fuente: `http://es.wikipedia.org/wiki/Test_de_Turing`, (Saygin et al., 2000)].

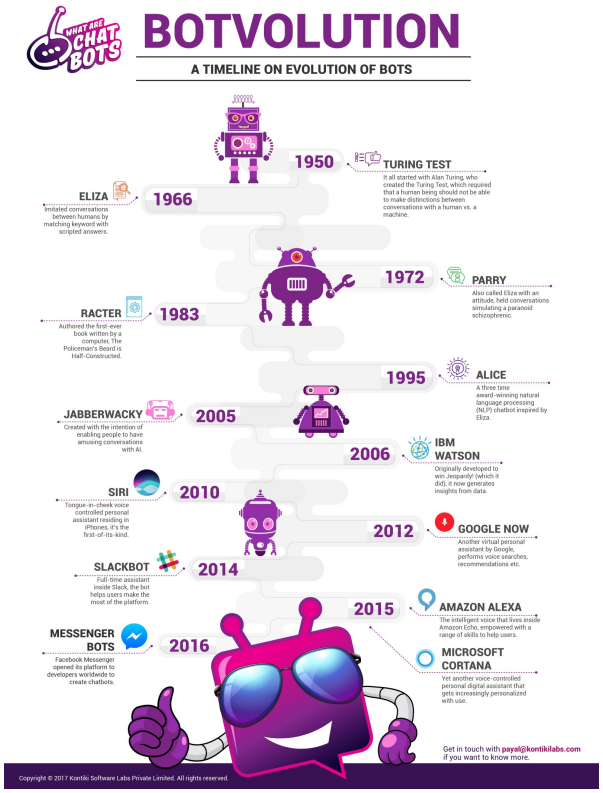


Figura 2.2: *Timeline* visual de la historia de los chatbots [Fuente: (Chakraborty, 2017).

Destacó el desarrollo de *Eliza*, un proyecto desarrollado por el Artificial Intelligence Laboratory en el MIT que intentaba emular a una psicóloga (Weizenbaum, 1966). Eliza busca palabras clave en la frase que el usuario escribe y responde utilizando una frase de su base de datos (ver Figura 2.3). En ocasiones sus respuestas se volvían incoherentes, cuando no entendía lo que había querido decirle.

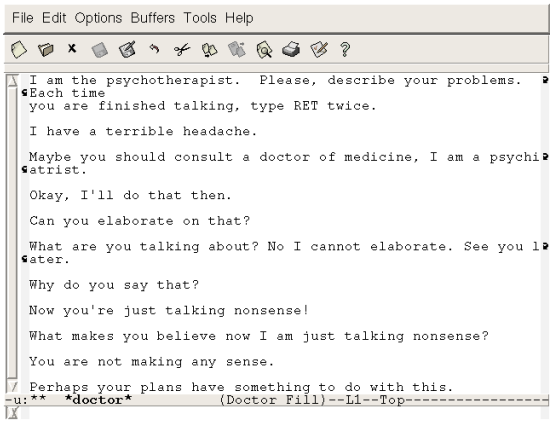


Figura 2.3: M-x doctor mode, an Eliza clone running in GNU Emacs [Fuente: `http://en.wikipedia.org/wiki/ELIZA`].

En 1995, inspirado por Eliza, comenzó el desarrollo del bot *Alice* (*Artificial Linguistic Internet Computer Entity*), capaz de recoger ejemplos de lenguaje natural (Wallace, 2009). El programa utiliza un esquema XML llamado AIML (Artificial Intelligence Markup Language) para especificar las reglas de conversación heurística. Los datos se podían encontrar organizados en categorías. Desde la página de Alice (`http://www.pandorabots.com/pandora/talk?botid=b8d616e35e36e881`), el usuario puede mantener una conversación con este programa inteligente y se simula una charla real en la que es difícil darse cuenta de que detrás existe un robot.

Con todos estos avances se consideró que los límites aún no se habían alcanzado. El reto ya se había propuesto desde comienzos de los 50 y muchos investigadores se unían a él: ordenadores capaces de batir al ser humano.

A principios del 2000 continuaba el desarrollo de la IA, se buscaba el desarrollo de robots con habilidades y personalidades o el aprendizaje en la navegación por un espacio. Entre ellos se puede destacar: la mascota robótica de Sony *AIBO*, Artificial Intelligence Robot (Fujita, 2001), o el innovador asistente robótico para el hogar *Roomba* (Forlizzi y DiSalvo, 2006). Diez años más tarde, Apple creó *Siri*, el primer asistente virtual para teléfonos (Siri, Apple, 2014), que surgió de la investigación del Artificial Intelligence Center de SRI International. A través del reconocimiento de voz y del procesamiento del lenguaje natural, Siri puede buscar la información solicitada en las aplicaciones del propio dispositivo o en Internet si no lo encuentra.

El mismo año IBM lanzó *Watson* al mercado (High, 2012). Este sistema inteligente es capaz de crear hipótesis y dar la respuesta con una mayor probabilidad.

Años más tarde, empresas como Microsoft y Amazon lanzan sus asistentes. *Cortana* (2014) se ha extendido en los smartphones, tabletas, computadoras y consolas de videojuego. Este asistente reconoce el lenguaje natural, puede aprender y adaptarse aprovechando sus bases con millones de datos (Microsoft Cortana, 2017). *Alexa* (2014) fue el primer asistente virtual creado por Amazon. Su principal característica es que se puede utilizar con el parlante inteligente llamado *Amazon Echo* (Ebling, 2016) y permite conocer información sobre clima, productos, compras, recordatorios, e inclusive video llamadas (Amazon Alexa Web Portal, 2017).

También las mensajerías móviles incluyen asistentes en sus plataformas (ver Figura 2.4). Por ejemplo, WeChat, KikMessenger o Telegram usan chatbots para realizar pagos en cafeterías o tiendas de ropa, para descargar música, reservar un taxi, comprar billetes de avión, etc. (Marupaka, 2018; Condé Nast Traveler, 2016; Claire by 30Secondstofly, 2016).

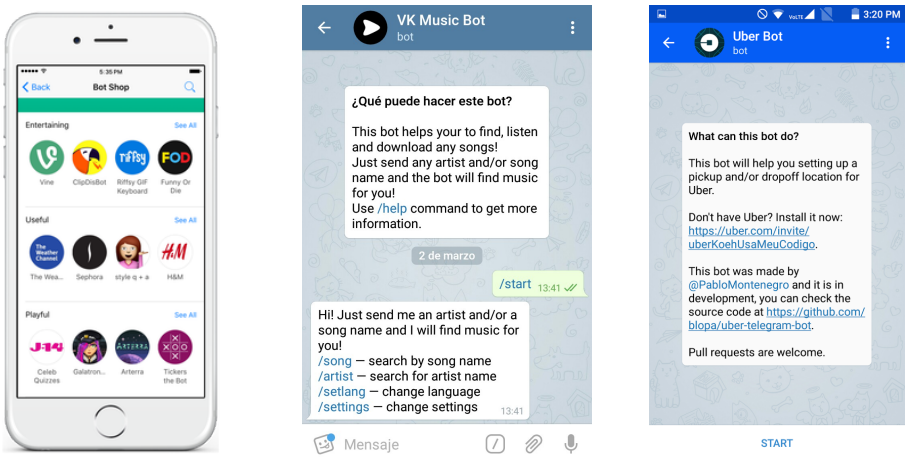


Figura 2.4: Chatbots de tiendas de ropa, música y reserva de taxis.

Por último cabe destacar la importancia que están tomando estas aplicaciones que utilizan IA a la hora de cubrir funciones de ayudas para el usuario. Como se puede ver en las páginas de Renfe (*Irene* es el asistente virtual de Renfe [`http://www.renfe.com/asistente.html`]), Carrefour, H&M (ver Figura 2.5), entre muchas otras.

Otros ejemplos de chatbots se pueden encontrar en:

- El *Bot-Hub Project* ofrece, de forma estructurada, descripciones y críticas de cientos de chatbots (`http://bot-hub.com`)
- *Kik bot shop*, una tienda virtual de bots (`http://bots.kik.com`)

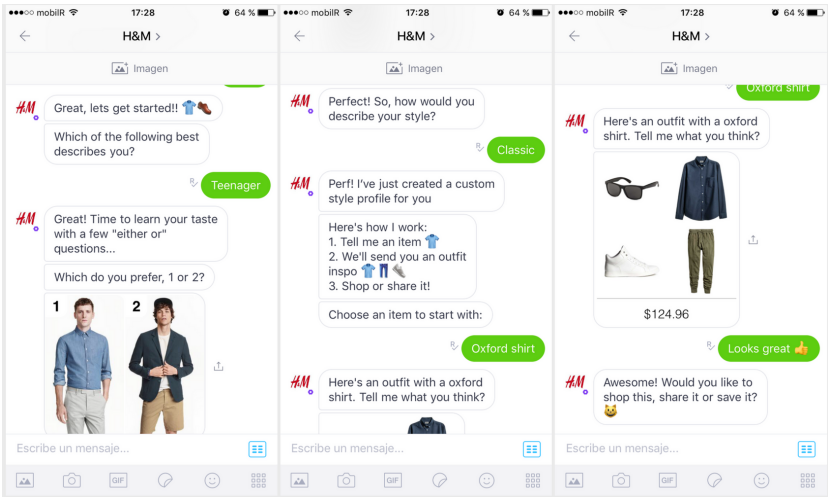


Figura 2.5: Chatbot de H&M [`http://www2.hm.com/es_es`].

- *BotList* es un directorio de los mejores bots para Messenger, Discord, etc. (`http://botlist.co`)
- *Bot Framework* es un catálogo de bots de Microsoft (`http://bots.botframework.com`)
- El directorio de chatbots ofrecido por la plataforma de bots *Chatfuel* (`http://chatfuel.com/bots`)

2.2. Chatbots en castellano

Como se ha mencionado anteriormente, los chatbots son aplicaciones que realizan una gran cantidad de cometidos: pueden buscar restaurantes, horarios de aviones, habitaciones de hotel, predicción del tiempo, entre muchas otras cosas.

Existen un gran número de chatbots para lengua inglesa. Aunque en lengua castellana aún no hay tantos, su número va creciendo. A continuación, se muestran los más populares y eficientes en castellano para distintos sectores:

- *El País Bot* es el primer bot conversacional del periódico El País. En función de las noticias suscritas, el usuario recibirá las noticias relacionadas. Propone una amplia gama de temas, que pueden ser tanto noticias como artículos de opinión o reportajes. Además se pueden buscar noticias recientes relacionadas con palabras clave, lo que te hace muy fácil ponerse al día (ver Figuras 2.6 y 2.7).
- Otro bot que está adquiriendo una gran cantidad de usuarios es *KittyBus* de la Empresa Municipal de Transportes de la ciudad de Madrid. Funciona sobre una aplicación Ruby on Rails con una base de datos

Figura 2.6: El País Bot.

Figura 2.7: Búsquedas en El País Bot.

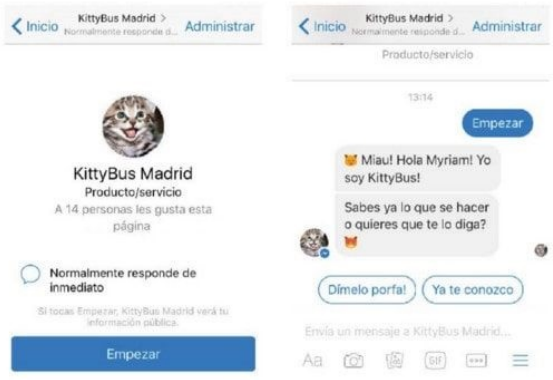


Figura 2.8: KittyBus.

SQLite. Permite obtener los datos precisos y a tiempo real gracias a su conexión con la API de la EMT Madrid (ver Figura 2.8).

Se trata de un gato que te dice cuanto tiempo tardará en llegar el autobús indicándole el número de parada o cuales son las paradas más cercanas a tu posición actual. Si el usuario se encuentra lejos de una parada, KittyBus también puede guardar las paradas favoritas para obtener la información en otro momento (ver Figura 2.9).

Figura 2.9: Conversación con KittyBus.

- *Lola Euroresidentes* es un chatbot que conoce el horóscopo. Está desarrollado por el grupo de trabajo IT&IS Open Space Technology. Está basado en el sistema api.ai de Google, una tecnología de creación de chatbots con aprendizaje automático. Desarrollan los corpus conversacionales temáticos sobre la base de árboles y grafos no dirigidos, con la finalidad de conseguir una mejor experiencia. Sus conversaciones abarcan todos las situaciones de la vida: trabajo, relaciones personales y familiares, etc.

- Si necesita hacer la compra y buscar/pedir los productos directamente a la tienda, *TiDi* es su asistente favorito. Se trata de un asistente del supermercado online `tudespensa.com` (ver Figuras 2.10 y 2.11). TiDi es muy simpática y dice ser prima de Siri. El usuario puede pedirle hablar directamente con el equipo humano si no encuentra lo que busca.

2.3. Plataformas de desarrollo

En esta sección se analizan las plataformas principales que existen para el desarrollo de chatbots de grandes empresas en el sector informático: IBM, Microsoft, Google, Facebook y Amazon (Ananthavel, 2018; Akiwatkar, 2017; Futurizable, 2017; Briz, 2017; Singh, 2017).

Figura 2.10: Chatbot TiDi del supermercado online `tudespensa.com`.

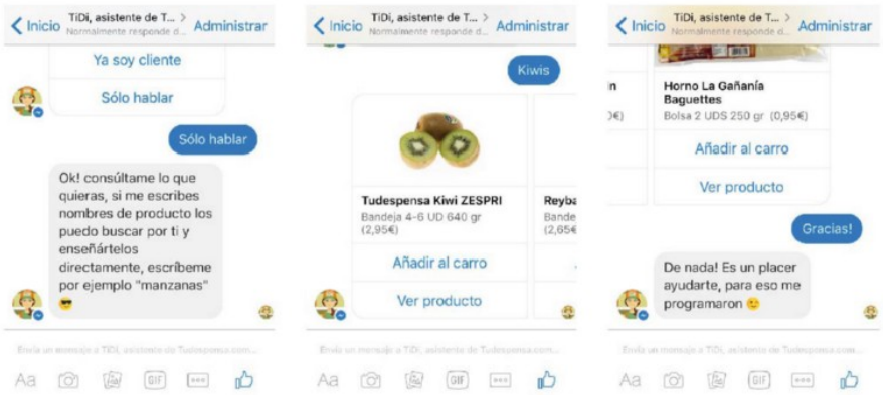


Figura 2.11: Consultas al chatbot TiDi.

Bluemix es la plataforma de Cloud Computing de **IBM**. En ella se pueden utilizar diferentes servicios y tecnologías en modalidad de alquiler o pago por uso. Dispone de una licencia gratuita y libre de uso para los estudiantes, que es la utilizada en este proyecto.

De todos los servicios que ofrece, se ha utilizado el servicio cognitivo *Watson Conversation* (ver Figura 2.12). Es una tecnología conversacional que permite diseñar nuestro propio chatbot (IBM, 2018b).

En el ámbito tecnológico, es normal que la persona se tenga que adaptar a la máquina, al ratón, al teclado, etc. Los sistemas cognitivos básicamente pretenden interaccionar como lo harían las personas. En primer lugar lee y entiende el lenguaje natural con sus matices y giros lingüísticos. Después consigue responder a preguntas complejas en pocos segundos, debido a su capacidad para analizar grandes cantidades de datos. Ante una situación

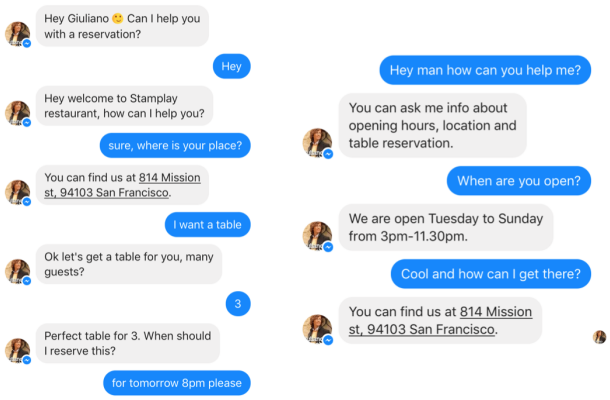


Figura 2.12: IBM Watson.

desconocida que no sepa resolver, formula hipótesis y escoge la respuesta que tenga un mayor nivel de confianza, mostrando su razonamiento.

Para poder responder a las preguntas, el sistema busca en una gran base de datos procedente de enciclopedias y archivos, para aprender tanto de la información como de los posibles contextos. La cantidad y calidad de los datos de los que se alimenta el sistema es muy importante. Esta selección de datos debe realizarse por un experto del área específica. Por último, el sistema aprende de cada experiencia y es entrenado por profesionales, así que cada vez es más inteligente.

IBM Watson tiene una gran capacidad de procesamiento, hasta 800 millones de páginas de información por segundo. Utiliza el razonamiento y aprendizaje automático (IBM, 2018a). Además tiene capacidad de procesamiento de lenguaje natural, generación de hipótesis, recopilación de pruebas masivas, análisis y emisión de recomendaciones con un alto grado de certeza.

Microsoft tampoco ha quedardo atrás en este movimiento con *Azure*, su plataforma de cloud. Proporciona un entorno integrado para el desarrollo de bots (conocido como *Azure Bot Service*) que está conectado con *Microsoft Bot Framework* y los *SDKs BotBuilder* (Microsoft, 2018).

Microsoft Bot Framework nos permitirá desarrollar bots que se comuniquen con los usuarios y realicen acciones automatizadas y desatendidas.

El SDK es una librería en Node.js y C# que se incorpora al proyecto del bot y se encarga de gestionar la conversación. Para incorporar comprensión del lenguaje natural el SDK se integra con la tecnología *Language Understanding Intelligence Service* (*LUIS*). Se compone de una serie de servicios alrededor del procesamiento de lenguaje natural como es el análisis lingüístico y ofrece un conjunto de herramientas que permite entrenar a la plataforma en modelos de conversación.

Otra de las funcionalidades que aporta es dar soporte al chat de habla, poder tener herramientas para trabajar con diálogos y definir cuál es el flujo de la conversación del bot. Dispone de herramientas para depurar nuestros bots y para poder visualizar los mensajes que se envían bot y servidor entre sí. De esta forma tiene un inspector de canales.

T-Bot es un chatbot desarrollado por la propia empresa Microsoft que utiliza esta tecnología (ver Figura 2.13). Es un bot con el que los usuarios pueden interactuar para obtener ayuda con el uso de *Microsoft Teams* y las respuestas obtenidas abarcan una amplia variedad de preguntas.

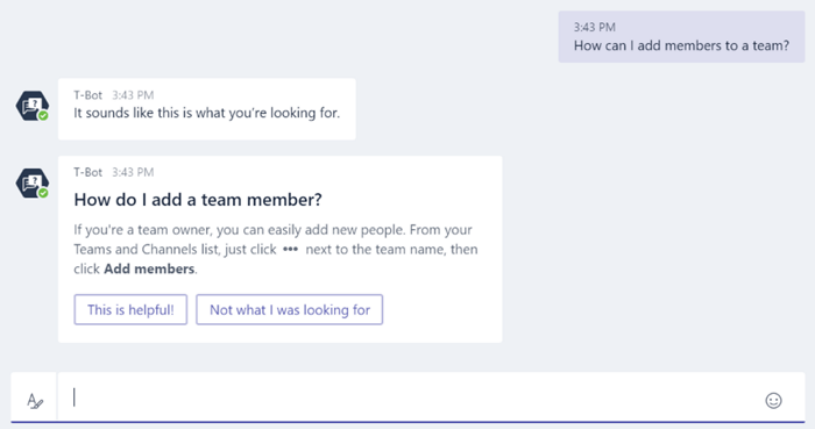


Figura 2.13: T-Bot de Microsoft.

Otra plataforma conversacional para bots es *Dialogflow*, anteriormente llamada *api.ai* (ver Figura 2.14). La plataforma es gratuita y fue comprada por **Google** en septiembre del 2016. Facilita la integración del agente con las acciones en *Google Assisstant*, que es un asistente personal y nos permite interaccionar por comandos de voz. Además permite la integración e importación/exportación con otras plataformas comunes como Alexa, Messenger, Skype y otras muchas (Dialogflow, 2018; Vinyals y Le, 2015; Business Insider, 2015).

Este agente está formado por intenciones y entidades. Las intenciones combinan las peticiones del usuario con acciones; mientras que las entidades agrupan palabras y sinónimos en frases de lenguaje natural. Además está formado por un contexto del tema en función de lo que el usuario dice, de la acción que toma el bot, etc.

Dialogflow nos permite establecer normas de reconocimiento natural del lenguaje y varias respuestas naturales. Un aspecto negativo de esta plataforma es que no admite enviar imágenes, solo se pueden incluir URLs.

Los SDKs están disponibles para varios lenguajes de programación (e.g., JavaScript, C++, Python y Java), frameworks de desarrollo para móviles (e.g., Cordova y Xamarin), y sistemas operativos (e.g., Android e iOS). Puede

trabajar con un gran número de idiomas, incluidos coreano, chino y ruso.

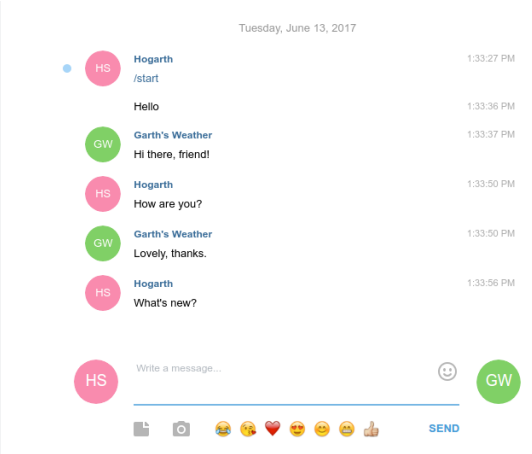


Figura 2.14: Dialogflow.

Facebook dispone de algunas de las principales plataformas sobre las que pueden funcionar los chatbots, como son Facebook Messenger y Whatsapp. Además en 2015 compró *wit.ai* un servicio por medio del cual los desarrolladores podían crear chatbots (wit.ai, 2018). Wit.ai está enfocado a usuarios con habilidades de programación y permite acceder con la cuenta de Github.

Esta plataforma permite el uso de entidades, intenciones, contextos y acciones para conseguir unas conversaciones más interactivas. Wit.ai genera conversaciones naturales y flexibles gracias a la incorporación del procesamiento de lenguaje natural. Permite también guardar la conversación como una API y así los datos se pueden descargar o compartir. Las Figuras 2.15 y 2.16 muestran dos ejemplos de aplicaciones desarrolladas con wit.ai.

Hay distintos clientes para otras plataformas como Node.js, Python, Ruby y HTTP API. Se puede utilizar con iOS, Android, Windows Phone, Raspberry Pi, Python, C y Rust.

Debido al gran auge que están teniendo los bots, **Amazon** también decidió sumarse a este movimiento. En su plataforma *Amazon Web Service* (AWS) se pueden encontrar una gran cantidad de servicios.

Entre ellos, *Amazon Lex* es un servicio para crear interfaces de conversación en cualquier aplicación con voz y texto. Ofrece funcionalidades avanzadas de deep learning para el reconocimiento automático del habla y la comprensión del lenguaje natural (Amazon, 2018).

Estas características permiten al desarrollador crear aplicaciones con interacciones de conversaciones realistas. Se pueden utilizar las tecnologías de Alexa, lo que permite crear con rapidez y facilidad bots conversacionales sofisticados con lenguaje natural. También permite publicar los chatbots de

Figura 2.15: Una sencilla aplicación del tiempo atmosférico con Wit.Ai.

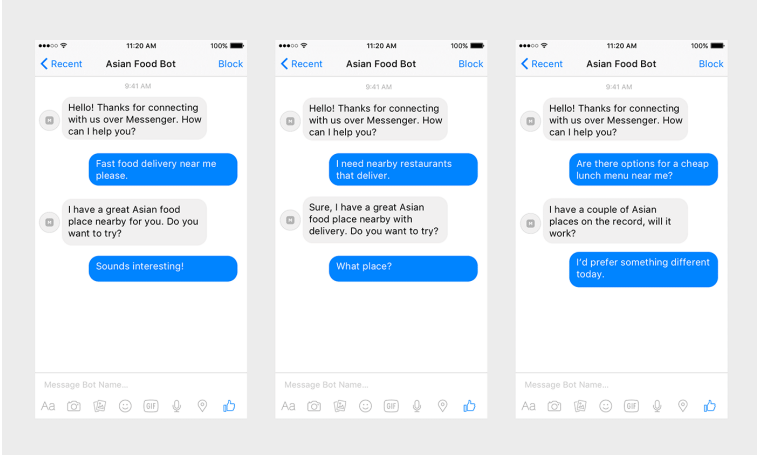


Figura 2.16: Aplicación con Wit.Ai con diálogos más complejos.

forma sencilla en dispositivos móviles y servicios de chat.

Alexa es una plataforma desarrollada por Amazon para empresas, que brinda a los equipos facilidades de organización tanto para los usuarios que lo componen como para sus dispositivos (Business Insider, 2016). Es capaz de realizar una gran cantidad de tareas de forma simple, como reproducir música o audiolibros, interactuar con la voz o proporcionar información meteorológica, de tráfico o noticias en tiempo real. Su interacción y comunicación sólo está disponible actualmente en inglés y alemán.

Alexa Skill kit es una colección de APIs, herramientas, documentación y ejemplos que facilitan el trabajo. Además permite la creación de capacidades para llegar a los clientes a través de una gran variedad de canales compatibles. La Figura 2.17 muestra un ejemplo —bot que ayuda a encargar flores—

desarrollado en esta plataforma.

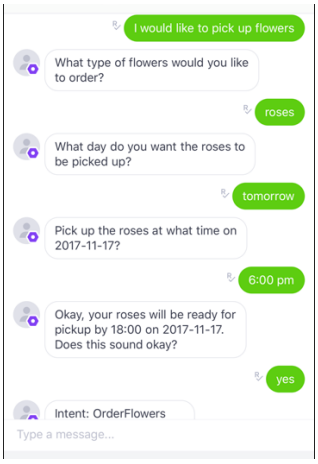


Figura 2.17: Bot desarrollado con AWS que ayuda a encargar flores.

A modo de resumen, se incluyen a continuación las Tablas 2.1 y 2.2 con las principales plataformas para construir chatbots. Otras plataformas importantes son: *Live Agent*, *Rasa NLU*, *Octane.ai*, *FlowXO*, *ManyChat*, *Gubshup*, *It's Alive* y *ChatScript* (Davydova, 2017).

Chatfuel

Nombre del bot	Plataforma	Características
IBM Watson	IBM Watson Conversation Service	Construido sobre una red neuronal. Tiene tres componentes principales: intenciones, entidades, diálogo
AgentBot	Tecnología de PLN propia de Aivo	Entiende el lenguaje natural; Memoria para mantener la coherencia en conversaciones largas; Recopila información del cliente para personalizar soluciones
Twyla	Plataforma AI propietaria	Aprende de los chats del agente/cliente; Combina aprendizaje automático y métodos basados en reglas; Contesta preguntas
Pypestream	Plataforma de mensajería inteligente propia	Pypestream utiliza un framework patentado de Pypes y Streams; Procesamiento de lenguaje natural y análisis de palabras clave
Digital-Genius	Herramienta de eficiencia de agentes de deep learning	Predice con AI los metadatos de los casos y sugiere las respuestas correctas para sus agentes; Aprende con AI de cada interacción de agente
Semantic Machines	Plataforma con AI conversacional patentada	Aprendizaje profundo; Reconocimiento de voz; Extrae la intención semántica; Utiliza *Conversation Engine* para generar lenguaje natural
Msg.ai	Plataforma de AI para el comercio conversacional	Modelos de intenciones y clasificaciones de tonos; Aprendizaje profundo

Tabla 2.1: Principales plataformas para construir chatbots (1/2) [Fuente: (Davydova, 2017)].

2.4. Técnicas existentes

El objetivo de un buen chatbot es conseguir que trabaje de forma inteligente. Existen dos modos de funcionamiento: el primero basado en un conjunto de reglas y un segundo modo más avanzado consiste en la aplicación de técnicas de Inteligencia Artificial. Con la definición de reglas, el chatbot es más limitado y sólo puede responder a unos comandos específicos.

En el ámbito de las ciencias de computación, se puede definir la IA como la capacidad que tiene una máquina de aprender, razonar y resolver problemas. Los pilares básicos en los que se basa la IA son (Nilsson, 2014):

- **Búsqueda del estado requerido en el conjunto de los estados producidos por las acciones posibles**

  Un estado es la representación de los elementos que describen al problema en un momento dado, es decir, la situación en la que se encuentra en un instante.

  Nilsson se refiere a la búsqueda del estado, como a la solución al problema planteado dentro del conjunto del espacio de estados. Siendo éste, un grafo cuyos nodos son todos los posibles estados que se pueden dar y las aristas son las transiciones entre ellos.

Nombre del bot	Plataforma	Características
wit.ai	-	Permite utilizar entidades, intenciones, contexto, comportamiento, proceso de lenguaje natural
Api.ai	-	Ajusta la consulta y la intención más adecuada en función de la información contenida en la intención y el modelo de aprendizaje automático del agente
Microsoft Bot Framework	-	Entiende la intención del usuario; Puede incorporar LUIS para la comprensión del lenguaje natural, APIs de Bing para búsquedas y Cortana para voz
LUIS	Microsoft Language Understanding Intelligent Service	Utiliza intenciones y entidades; Aprendizaje activo; LUIS devuelve JSON fácil de usar
Pandorabots	-	AIML; Incluye A.L.I.C.E.
ChatterBot	-	Selecciona la respuesta con mejor coincidencia
Rebot.me	-	Permite crear un chatbot y entrenarlo con preguntas y sus respuestas correspondientes; Ofrece un registro de chat con los resultados de conversaciones anteriores
Reply.ai	Visual Bot builder	Aprovecha los motores de PLN de wit.ai y api.ai para sus casos de uso avanzado
KITT.AI	ChatFlow	Detección de palabras clave (no se requiere internet); PLN y análisis semántico; Redes neuronales

Tabla 2.2: Principales plataformas para construir chatbots (2/2) [Fuente: (Davydova, 2017)].

- **Algoritmos genéticos**

  Un algoritmo genético es un método adaptativo que puede usarse para la resolución de problemas de búsqueda y optimización. Están basados en el proceso genético de los organismos vivos. Las poblaciones evolucionan en la naturaleza a lo largo de las generaciones acorde con los principios de la selección natural y supervivencia de los más fuertes postulados por Darwin. Los algoritmos genéticos son procesos de búsqueda heurística que simulan la selección natural. Usan métodos tales como la mutación y el cruzamiento para generar nuevas clases que puedan ofrecer una buena solución al problema.

  Con la imitación de este proceso, los algoritmos son capaces de crear soluciones para problemas del mundo real. La evolución de dichas soluciones hacia valores óptimos depende de la codificación de las mismas. Es decir, consiste en una función matemática o rutina de software que toma como entradas a los ejemplares y devuelve como salidas ejemplares que deben generar descendencia para una nueva generación.

- **Redes neuronales**

  Son un paradigma de aprendizaje automático cuya función es emular al cerebro humano inspirado en las neuronas del sistema nervioso. Se

trata de un sistema de enlaces de neuronas que colaboran entre sí para producir un estímulo de salida. Las conexiones tienen pesos numéricos que se adaptan según la experiencia o entrenamiento. De esta manera, las redes neuronales se adaptan a un impulso y son capaces de aprender.

La importancia de las redes neuronales decayó durante un tiempo, pero volvió a resurgir a finales de la década de 2000 con la aparición del aprendizaje profundo.

- **Razonamiento mediante lógica formal análoga al pensamiento abstracto humano**

  El razonamiento que puede tener cualquier persona es una de las tareas más difíciles de modelar en un ordenador. Se debe a que los razonamientos no suelen ser exactos y las conclusiones no están basadas en unas reglas binarias.

Se pueden distinguir entre dos corrientes según el método de desarrollo: simbolistas y conexionistas (Russell y Norvig, 2016). La corriente *simbolista* se apoya en los principios planteados por Simon y Newell, que defienden que los procesos inteligentes deben poder expresarse mediante operaciones sobre símbolos. Los *conexionistas*, inspirados por McCulloch y Pitts, simulan la inteligencia humana a través de las conexiones que se establecen entre las unidades, de la misma forma que las neuronas.

Las redes neuronales artificiales son apropiadas para aplicaciones como el reconocimiento de patrones o el manejo de voz, caracterizadas por la existencia de ruido y por la dificultad de especificar el dominio. Mientras que el simbolismo ha resultado más útil cuando la información disponible está más formalizada.

Para profundizar más, se comentan dos de las ramas que estudia la Inteligencia Artificial: el procesamiento del lenguaje natural y el aprendizaje automático. Se incluyen también unas nociones y características del aprendizaje profundo.

1. **Procesamiento del lenguaje natural**

   Es la disciplina encargada de producir sistemas informáticos eficientes que posibilitan una comunicación, por medio de voz o texto, entre las personas y las máquinas a través del lenguaje natural. Además de la tarea de comprensión y generación del lenguaje, se centran en los aspectos cognitivos humanos y en la organización de la memoria.

   Las principales aplicaciones son, entre otras, la traducción automática de textos y análisis del sentimiento, realización de resúmenes, clasificación de documentos por categorías, recuperación y extracción de la información, sistemas conversacionales o dar respuestas automáticas a preguntas.

El procesamiento del lenguaje natural requiere varias tareas (Carbonell, 1994), que se pueden descomponer en:

- **Análisis morfológico**: El análisis de las palabras para extraer raíces, rasgos flexivos, unidades léxicas compuestas y otros fenómenos.
- **Análisis sintáctico**: El análisis de la estructura sintáctica de la frase mediante una gramática de la lengua en cuestión.
- **Análisis semántico**: La extracción del significado de la frase y la resolución de ambigüedades léxicas y estructurales.
- **Análisis pragmático**: El análisis del texto más allá de los límites de la frase, por ejemplo, para determinar los antecedentes referenciales de los pronombres.
- **Planificación de la frase**: Estructurar cada frase del texto con el fin de expresar el significado adecuado.
- **Generación de la frase**: La generación de la cadena lineal de palabras a partir de la estructura general de la frase, con sus correspondientes flexiones, concordancias y restantes fenómenos sintácticos y morfológicos.

Tratar la lengua computacionalmente requiere un proceso de modelización. En la década de los 50 surgieron unos primeros modelos que pretendían reflejar la estructura lógica del lenguaje. La mayoría de estos sistemas se basaban en modelos lógicos, en los que los lingüistas escriben reglas de reconocimiento de patrones, empleando un formalismo gramatical. Estas reglas, junto con la información almacenada necesaria, definen los patrones que hay que reconocer para resolver las tareas (Moreno, 2017). A finales de 1980, hubo una revolución con la introducción de algoritmos de aprendizaje automático para el procesamiento del lenguaje.

2. **Aprendizaje automático**

   Rama de la IA cuyo objetivo es desarrollar técnicas que permiten aprender a los ordenadores. La aproximación que se utiliza en este ámbito consiste en modelos probabilísticos del lenguaje natural basados en datos. Los lingüistas recogen colecciones de ejemplos y datos. A partir de ellos se calculan las frecuencias entre distintas unidades y la probabilidad de aparecer en determinados contextos.

   De esta forma se podrá predecir cual es la siguiente unidad en un contexto dado, sin necesidad de recurrir a reglas explícitas. Este es el paradigma del aprendizaje automático, donde se pueden inferir posibles respuestas en función de los datos observados.

Tiene una amplia gama de aplicaciones, incluyendo por ejemplo motores de búsqueda, diagnósticos médicos, reconocimiento del habla y del lenguaje escrito, entre otras.

Podemos distinguir una taxonomía en función de la salida los distintos algoritmos del aprendizaje automático (Ng et al., 2013):

- *Aprendizaje supervisado*: El algoritmo establece una correspondencia entre las entradas y salidas del sistema. El *profesor* suministra ejemplos al sistema. Este tipo de aprendizaje puede llegar a ser muy útil en problemas de investigación bioinformática, biológica y biología computacional.
- *Aprendizaje no supervisado*: No hay *profesor* que verifica los ejemplos. El objetivo es descubrir patrones en el conjunto de entrenamiento que permitan agrupar y diferenciar unos ejemplos de otros.
- *Aprendizaje por refuerzo*: Es un aprendizaje semi-supervisado, donde se combinan los dos anteriores. El sistema recibe algún tipo de recompensa cada vez que produce una respuesta, ajustando su comportamiento en función de dicha recompensa.

En función del tipo de enfoque, el tipo de *machine learning* utilizado puede ser: árboles de decisión, reglas de asociación, algoritmos genéticos, redes neuronales artificiales, máquinas de vectores de soporte, algoritmos de agrupamientos y redes bayesianas.

3. **Deep learning**

   El *deep learning* consiste en un conjunto de algoritmos dentro del aprendizaje automático y con aprendizaje no supervisado (García Moreno, 2018). Mientras que los algoritmos tradicionales para el *machine learning* son lineales, en el aprendizaje profundo los algoritmos se apilan en una jerarquía de creciente complejidad y abstracción, donde se intentan modelar abstracciones de alto nivel de los datos (Rodríguez, 2017).

   El aprendizaje profundo representa un acercamiento al modo de funcionamiento del sistema nervioso de un humano. En lugar de enseñarle una lista de reglas para resolver un problema, proporcionamos un modelo que pueda evaluar ejemplos y una pequeña colección de instrucciones para modificarlo cuando se produzcan errores. Con el tiempo, esperamos que esos modelos sean capaces de solucionar el problema de forma precisa, gracias a su capacidad de extraer patrones.

   Aunque existen diversas técnicas, una de las más comunes es la simulación de sistemas de redes artificiales de neuronas. Se caracterizan por tener una arquitectura en capas, donde cada capa aprende patrones

más complejos según la profundidad en la que se encuentre. Las capas forman una jerarquía de características desde un nivel de abstracción más bajo a uno más alto.

Existen redes neuronales que tienen esta característica como las redes convolucionales o redes recurrentes. Sin embargo, existen otras que no se consideran profundas como el perceptrón, que no se considera deep learning al tener una única capa.

2.5. Desafíos y problemas

Los chatbots se encuentran aún en sus primeras etapas de desarrollo, pero se puede imaginar en qué convertirán en el futuro. Hoy en día ya tienen un enorme potencial de beneficios. Con los años se están convirtiendo en una amenaza para los buscadores.

Aunque no todo son ventajas, el problema principal en la mayoría de los bots es que son poco inteligentes o útiles. Su desarrollo es muy fácil sin necesitar nociones de programador, pero la calidad de sus funcionalidades empeora notablemente (Mas Digital, 2017).

A lo largo de la historia las marcas/empresas han intentado acercarse a los clientes con el objetivo de convertirse en sus amigos y, a través de los asistentes, intentan involucrarlos en sus negocios. A medida que esta tecnología avance, deberán proporcionar confianza a los usuarios. Los usuarios buscan una asistencia personalizada, comportamientos creativos y que sean capaces de entender la sensibilidad que podría estar sintiendo el usuario humano.

Sin embargo, hay que ser cuidadoso cuando se imita el comportamiento humano. Un ejemplo es el chatbot *Tay* de Microsoft que fue un fracaso (ver Figura 2.18). Se diseñó para imitar los patrones de lenguaje de una adolescente americana y para aprender de las interacciones con los usuarios de Twitter (TayTweets, 2017). Aunque tenía temas censurados, el chatbot enviaba mensajes racistas y cargados de contenido sexual. Microsoft tuvo que retirar el servicio en tan solo 16 horas.

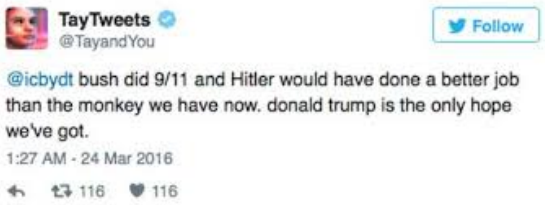


Figura 2.18: Chatbot *Tay*.

Uno de los problemas centrales que podemos encontrarnos en la manipulación del lenguaje natural es la ambigüedad en el idioma. Puede ser reflejado

por distintos niveles: léxico, referencial y pragmático (ver Figura 2.19).

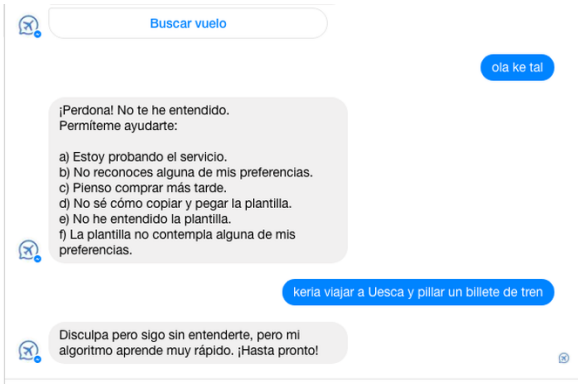


Figura 2.19: Lenguaje utilizado por los *milenials*.

En el *nivel léxico*, una palabra puede tener varios significados y la selección del significado apropiado se debe deducir del contexto adicional. Muchas investigaciones estudian formas de resolverlo, aplicando métodos mediante diccionarios, gramáticas, bases de conocimiento y correlaciones estadísticas.

En el *nivel referencial* es necesario conocer el hilo de la conversación o entender la entidad a la que se refiere, para tener en cuenta las anáforas o catáforas.

Por último, en el *nivel pragmático*, en muchas ocasiones las frases no expresan lo que literalmente dicen. Existen elementos como la ironía que forman parte de la interpretación.

Hay que destacar que la tecnología ha cambiado la forma de comunicación y los *milenials* no utilizan el lenguaje como la gente "mayor". En sus conversaciones es cotidiano encontrarse con abreviaturas, contradicciones o faltas de ortografía (Sierra, 2017). Además pueden no leer los mensajes y responder sin seguir el hilo de la conversación, lo que dificulta la comprensión con los chatbots.

Capítulo 3

Contribución

Para afrontar este proyecto, a grandes rasgos se pueden distinguir las siguientes fases:

- Investigación de técnicas y tecnologías
- Implementación de un prototipo
- Análisis de los resultados y conclusiones

La primera fase se ha expuesto en los capítulos anteriores y, en este capítulo, se detallará la aplicación que se ha implementado. En primer lugar se introducen las funcionalidades del recomendador de museos de Madrid que se desea implementar. En el Capítulo 4 se puede encontrar el primer prototipo: *Asistente Turístico de Madrid*. Después se explicará la arquitectura que se ha utilizado y se finalizará con un ejemplo de funcionamiento.

3.1. Descripción de la funcionalidad del sistema

El objetivo final es diseñar una arquitectura de un bot recomendador de museos que esté integrado en un canal para facilitar la comunicación con el usuario.

Como la versión anterior había sido muy limitada, se quiere ahora utilizar una fuente externa a Watson y poder realizar consultas utilizando los parámetros que el bot identifica (entidades, intenciones, variables, etc.). De otro modo, tendríamos que gobernar el hilo de la conversación desde el bot o tener un número de ramas que no consideramos escalable.

Se espera implementar un bot con el que, mediante una aplicación de mensajería a través de una simple conversación, pueda recomendarnos un museo al que poder ir en base a unas características.

La recomendación del bot se puede realizar considerando distintos atributos:

- Por un lado, se puede solicitar la recomendación en función del **precio**, distinguiendo entre museos *baratos*, *medios* y *caros*.
- Existe también la posibilidad de solicitar un museo en función de la **localización**. Se distinguen las siguientes zonas: *norte*, *sur*, *este*, *oeste* y *centro de Madrid*. Clasificando sus barrios entre estas clases para poder realizar la búsqueda del museo.
- Por último se puede distinguir por **tipos de museos**. Existe un amplio repertorio de categorías, entre las que hemos definido están: *antropológico*, *arqueológico*, *arte contemporáneo*, *artes decorativas*, *bellas artes*, *ciencias naturales*, *científico tecnológico*, *histórico y etnográfico*, *marítimo y naval*, *militar* y *musical*.

3.2. Arquitectura para el diseño

En esta sección se incluye una pequeña introducción de las tecnologías que han sido escogidas para el prototipo final y se describe la infraestructura del resultado final de la aplicación del bot con cada una de las siguientes tecnologías:

- IBM Watson
- Telegram (Front-end)
- Python (Back-end)
- Mongodb (Storage)

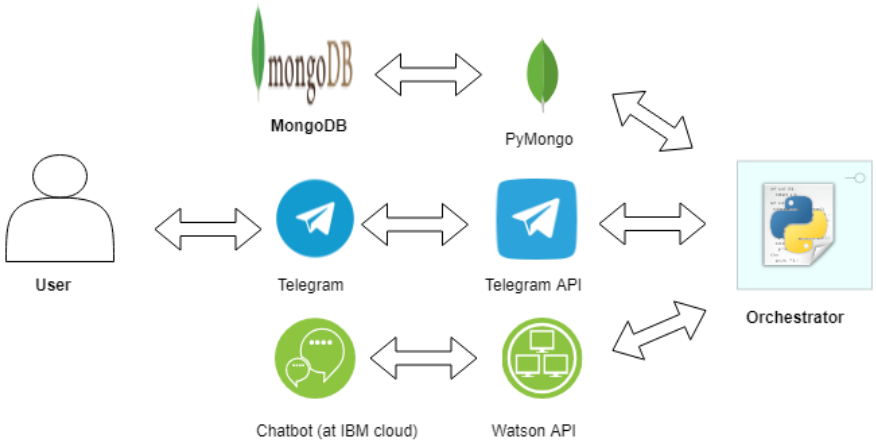


Figura 3.1: Arquitectura de la aplicación.

La solución de esta aplicación consiste en implementar un orquestador, que es básicamente un cliente del bot. Esta solución se llevó a cabo inicialmente para el primer prototipo de una forma más simple. Como el proceso es similar, en la Figura 3.1 se incluye sólo el desarrollo del último por ser más completo.

Primero se realiza una breve introducción de las tecnologías que han sido escogidas, resaltando las características útiles para el proyecto.

Python es un lenguaje de programación interpretado y muy potente (Python Software Foundation, 2017; Cannon, 2018; Van Rossum y Drake, 2011). Gracias a esta característica, funciona en cualquier sistema que integre su intérprete. Python es multiplataforma, por lo que se puede ejecutar en diferentes sistemas operativos como Linux o Windows. No sólo es multiplataforma y multiparadigma, sino que además cuenta con numerosos frameworks que facilitan el desarrollo y la conexión entre distintas tecnologías (Guttag, 2013; Kuhlman, 2009).

IBM Watson ofrece una API para cada bot, que permite identificarlo con el `workspaceId`, `clientId` y `password`. Además gestiona la comunicación entre el usuario y el bot, recibiendo toda la información asociada al procesamiento del lenguaje reconocida por el mismo (IBM, 2018a).

Telegram proporciona una interfaz para interactuar con los usuarios (Telegram group, 2013; Malizia, 2016). Está creado sobre una serie de servidores distribuidos que, para comunicarse, utilizan un protocolo propio llamado *MTProto*. Este protocolo ofrece una mejora en la seguridad y en el envío de mensajes, vídeos e imágenes (Wikipedia, 2016). El API con el que se interactúa está diseñado para ocultar toda la sección del protocolo de cifrado. De esta forma, su manejo resultará más natural y sencillo (ver Figura 3.2).

Para el almacenar el conocimiento, se ha utilizado **MongoDB** (Membrey et al., 2011; Banker, 2011). Es una base de datos *NoSQL* orientada a documentos. Esto quiere decir que, en lugar de guardar los datos en registros, se almacenan en documentos con formato *BSON*, que es una representación binaria de *JSON* (Chodorow, 2013). Además, existe una librería de Python conocida como *Pymongo*, para trabajar con MongoDB y poder realizar el tratamiento con las otras plataformas.

Para la infraestructura, se explicará primero la arquitectura sin la integración con Telegram, que corresponde al datapath sin canal.

La comunicación entre el usuario y el chatbot se realiza a través del orquestador desarrollado en Python. La API a la que nos referimos en esta sección del documento es de tipo REST y la información se intercambia en formato JSON.

Al comenzar la ejecución del programa se leen las credenciales y los identificadores del workspace. Después éstas son enviadas a la API de Watson para iniciar la conversación con nuestro bot.

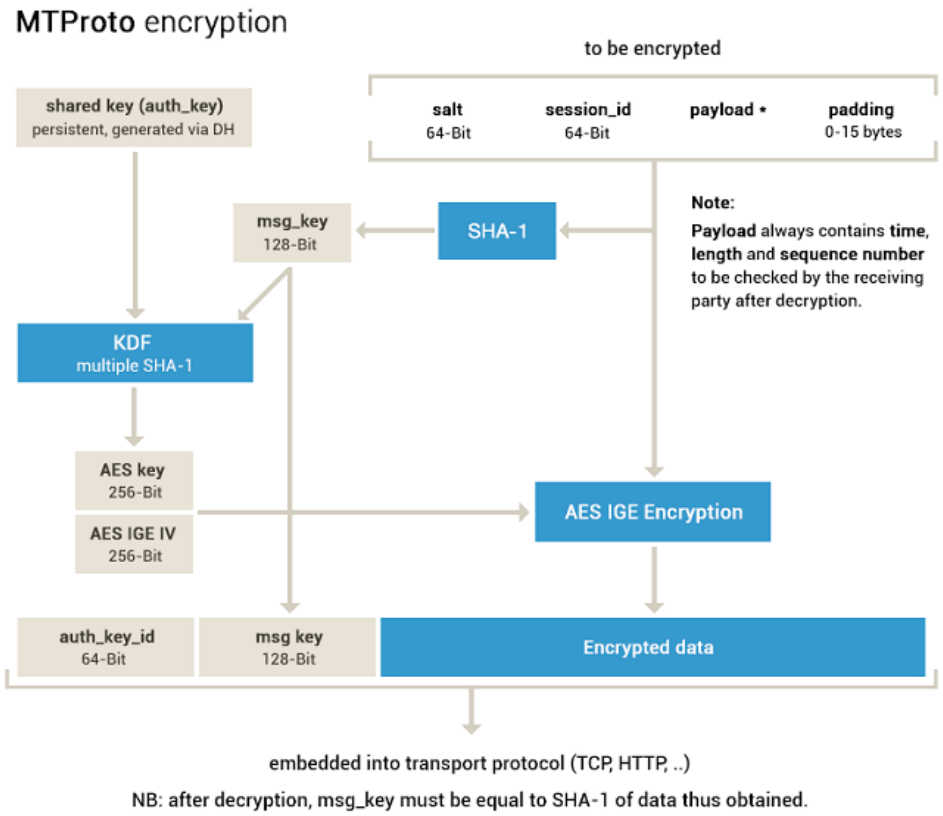


Figura 3.2: Protocolo de encriptación de Telegram [Fuente: (Telegram, 2018)].

El diálogo ha sido diseñado para que cuando se inicie la conversación con el usuario, el orquestador reciba el texto de ese diálogo, se lo muestre al usuario y permanezca a la espera de alguna nueva respuesta. Además recibe un contexto inicialmente vacío que se incluirá en cada respuesta.

Cada respuesta que el usuario introduce en la interfaz del orquestador se retransmite al bot por el orquestador a través de la API. Se envía el texto del mensaje y el contexto que mencionamos en el punto anterior, replicando el bot en cada petición que se realice.

Cada vez que el bot responde al usuario, el orquestador recibe un objeto con el texto de la respuesta, y un campo con la información de la entidades, intenciones y variables que éste ha detectado en el diálogo a este momento.

A parte de replicar estas variables al bot, para que conozca en qué rama y nodo estamos, las procesaremos en la lógica del orquestador para poder realizar la consulta en la base de conocimiento de Mongo y así mostrar la recomendación al usuario.

Se repetirá el proceso anterior hasta que la aplicación haya recibido la información suficiente para mostrar al usuario el resultado de alguna consulta en su fuente de conocimiento. Una vez realizada la consulta se proporciona su resultado al usuario y se procede a finalizar la conversación.

Este proceso no incluye la integración con Telegram. Para esta parte, se amplió el orquestador con una librería que proporciona Python. Consiste en una API para bots que pone a disposición tipos y métodos, dando la posibilidad de manejar con mayor facilidad la lógica del programa.

Finalmente, para poder utilizarlo con este canal, se creó un bot en Telegram identificado como **@ArtMadridBot** o **ArtBot**. A través de este canal y buscando el bot por alguno de esos nombres, se iniciará una conversación cuyo objetivo es la recomendación de un museo en Madrid.

3.3. Ejemplo de uso

A continuación se muestran los pasos a seguir para conectarse al chatbot. Para poder utilizarlo es necesario tener instalada la aplicación *Telegram*, que funcionará de canal entre el chatbot y el usuario.

Una vez descargado, en el buscador de la pantalla principal habrá que introducir `@ArtMadridBot`. La Figura 3.3 muestra como se realiza la búsqueda.

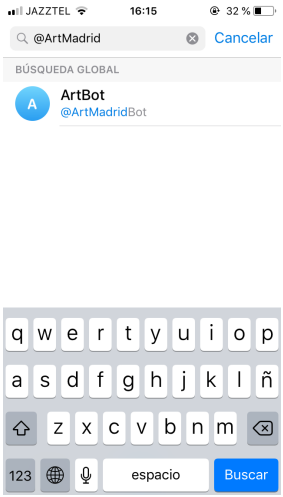


Figura 3.3: Búsqueda del bot en Telegram.

Después se debe empezar una conversación. Cuando ya se ha entrado en el chat, sólo se tiene que iniciar la conversación para comenzar a hablarle. La Figura 3.4 muestra el inicio de la conversación.

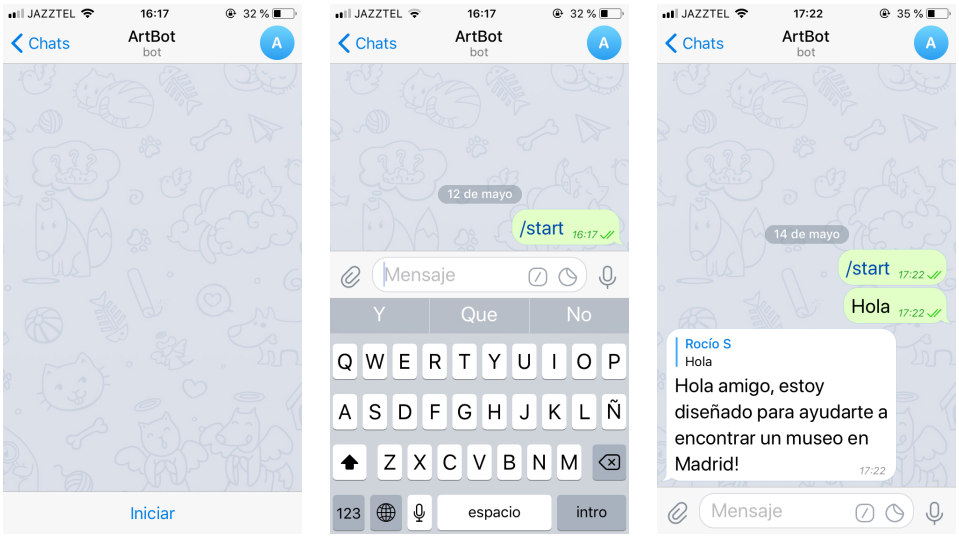


Figura 3.4: Inicio de conversación en Telegram.

`ArtBot` también dispone de un menú de ayuda para poder entender su uso (ver Figura 3.5).

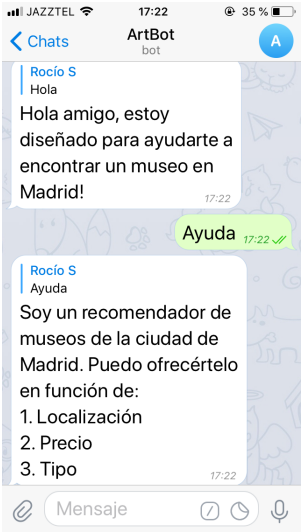


Figura 3.5: Menú de ayuda en Telegram.

Por último se muestran los distintos tipos de recomendación que soporta este chatbot (ver Figura 3.6).

Figura 3.6: Recomendaciones del bot por precio, lugar y tipo.

Capítulo 4

Fases del proyecto

En este capítulo se comentan con detalle las fases por las que ha evolucionado el proyecto desde el análisis inicial hasta la elaboración del prototipo final.

4.1. Fase 1: Análisis del proyecto e investigación de tecnologías existentes

El primer paso del proyecto ha sido el análisis general de la propuesta, así como la adquisición de conocimiento en los temas relacionados. La búsqueda de la tecnología necesaria para la realización del prototipo ha sido importante desde los comienzos del proyecto.

Otro aspecto que se analizó e investigó fue el funcionamiento de los chatbots. Se recopiló información sobre la inteligencia que se esconde detrás de una simple conversación. Llama la atención el especial interés que se muestra últimamente por los bots conversacionales.

El chatbot que se busca desarrollar debe de ser fácil, intuitivo y agradable de usar para el usuario final, pues es quien utilizará luego el sistema. Debe ser efectivo en el dominio que se haya especificado. Por ello se investigaron campos que se pudieran cubrir y resultaran de interés para los usuarios.

El proyecto se ha dividido en una serie de fases, que se enumeran a continuación:

- **Fase 1**: Análisis del proyecto e investigación de tecnologías existentes
- **Fase 2**: Diseño y desarrollo del bot
- **Fase 3**: Instalación y ejecución

Una vez acabada esta fase de análisis, cuyos resultados se ven reflejados en los primeros capítulos de esta memoria, la siguiente fase consistió en la

elección de la herramienta a utilizar, la definición de la funcionalidad del bot y su implementación.

4.2. Fase 2: Diseño y desarrollo del bot

En esta fase se presentan los pasos que se siguieron hasta finalmente conseguir el prototipo final. Se detallarán la herramienta propuesta y sus ventajas. También se describe la funcionalidad del bot y los problemas que surgieron durante el desarrollo.

4.2.1. Fase 2.1: Tecnología - Microsoft Bot Framework

En un primer momento se tomó la decisión de implementar el bot con la herramienta que proporciona Microsoft.

Azure es un conjunto integral de servicios en la nube que los desarrolladores utilizan para crear, implementar y administrar aplicaciones (Microsoft, 2018). Una de las herramientas que posee es *Azure Bot Service*, que es un servicio de bots inteligentes que se escala a petición y permite a los desarrolladores crear interfaces de conversación para distintos escenarios. Permite además acelerar el desarrollo de bots con cinco plantillas que se pueden elegir en el momento de la creación y se pueden modificar/personalizar aún más en un entorno de desarrollo integrado, como *Visual Studio*.

Los beneficios que encontré en esta herramienta fueron varios. Por ejemplo, la facilidad de conectividad con canales, es decir, la conexión entre el *Bot Framework* y las aplicaciones de comunicación como Skype y Facebook Messenger, entre otras (ver Figura 4.1). El *Bot Connector* es el encargado de conectar un bot a uno o más canales y maneja el intercambio de mensajes entre ellos. De esta forma controla la conversación proporcionando una abstracción de la complejidad de cada plataforma.

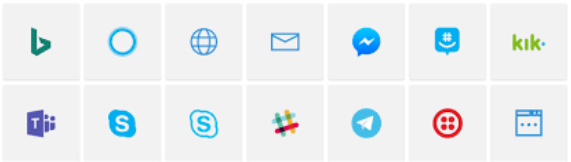


Figura 4.1: Diferentes canales disponibles para conexión con el bot.

Otra ventaja que ofrece Azure es la posibilidad de escribir un bot, conectarlo, probarlo, implementarlo y administrarlo desde su navegador web sin necesidad de un editor o control de fuente por separado.

Tomada la decisión de la herramienta a utilizar, en esta siguiente fase fue necesario preparar el entorno y, una vez instalado, diseñar la funcionalidad del chatbot.

4.2.1.1. Fase 2.1.1: Preparación del entorno

El primer paso fue preparar el entorno necesario para esta herramienta, que ofrece la posibilidad de utilizarse a través de la web o con un entorno de desarrollo. Luego se preparó el entorno para utilizarse en ambas situaciones.

Se instaló el entorno de Visual Studio 2017 junto con el emulador para realizar las pruebas. Además se creó una cuenta en Azure para obtener un cloud vinculado a una cuenta local y las cuentas asociadas a los recursos para el bot.

Una vez iniciados ambos, comencé con la creación de un chatbot sencillo *Hello Word*, utilizando unas plantillas básicas que proporcionaba el entorno. Consiguiendo su funcionamiento local a través del emulador. Una vez conseguido el funcionamiento local con el entorno, me propuse conectar mi trabajo con la cuenta de Azure, para así poder probar las herramientas que ofrece. Aquí fue donde surgió el problema.

4.2.1.2. Fase 2.1.2: Estudio y solución del problema

Una vez creadas las cuentas e instalados los entornos, se implementó un bot que funcionó correctamente de forma local (ver Figura 4.2). Después quería ejecutarlo dentro de *Azure bot service* ya que ofrece distintos servicios como:

- **Build**: Proporciona opciones para realizar cambios en el bot.
- **Test in Web Chat**: Permite utilizar el control integrado de *Web Chat* para ayudar a probar el bot de una forma rápida.
- **Analytics**: Si el análisis está activado para el bot, se pueden ver los datos analíticos que *Application Insights* ha recopilado para él.
- **Channels**: Configuración de los canales que puede usar un bot para comunicarse con los usuarios.
- **Settings**: Sección de configuraciones del perfil de bot como, por ejemplo, el nombre de visualización, el análisis y el punto final de mensajería.
- **Speech priming**: Gestión de las conexiones entre la aplicación LUIS y el servicio *Bing Speech*.
- **Bot Service pricing**: Gestión del nivel de fijación de precios para el servicio del bot.

La publicación del proyecto desde Visual Studio al portal de Azure pareció realizarse con éxito, porque no se generó ningún mensaje de error. El problema vino asociado en el propio portal, que no se actualizaba con la

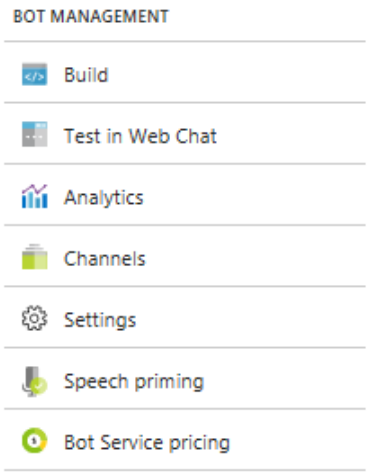


Figura 4.2: Herramientas de gestión del bot.

información del proyecto publicado. Me dediqué a recopilar documentación fiable de las herramientas y consultar foros de MS Azure.

Concretamente el error se producía en la pantalla de *Web App Bot*, al probar la ventana *Test in Web Chat*, pues permanecía de forma indefina el mensaje `Waiting for bot to be ready`. Aunque en un foro se comentaba que a veces solía pasar esto y bastaba con recargar la página, nunca conseguí acceder a él.

También a la hora de probar la conexión a un canal, por ejemplo, con *Web Chat*, se encontró en un estado de error apareciendo el siguiente mensaje: `There was an error sending this message to your bot: HTTP status code InternalServerError`.

Intentar solucionar estos problemas supuso bastante dedicación de tiempo. Finalmente, al no tener éxito en los intentos de resolución del problema y en la búsqueda de otras alternativas con Azure, opté por volver a revisar otras tecnologías que pudieran adaptarse al desarrollo del proyecto.

Otra de las opciones que parecía interesante era *IBM Watson*. La Universidad Complutense de Madrid ofrecía una licencia para su uso como estudiante.

IBM Watson permite a los usuarios construir flujos de conversación de lenguaje natural usando un entorno de usuario gráfico con técnicas de machine learning. Además, para poder automatizar las interacciones con los usuarios finales, facilita la implementación proporcionando a los desarrolladores una interfaz de lenguaje natural para aplicaciones.

También combina el aprendizaje automático con comprensión del lenguaje natural y herramientas de diálogo integradas. Por tanto, estas características determinaron que escogiera IBM Watson para construir el prototipo.

4.2.2. Fase 2.2: Tecnología - IBM Watson

Como mencionamos anteriormente, Watson es un sistema desarrollado por IBM que, haciendo uso de su inteligencia, es capaz de responder a preguntas e interactuar mediante el lenguaje natural. También ofrece ayudas para el desarrollo del chatbot y una serie de herramientas como analíticas, recomendaciones o protección de datos.

En esta fase se explica el proceso que se siguió después de la decisión de utilizar el servicio *Watson Conversation*. En primer lugar, se tratará la configuración del entorno, la configuración de los diálogos y la forma en que se entrena el sistema. Después se definirán tanto el dominio como los diálogos. Por último se desarrollará una aplicación que, usando la API de Watson, nos permitirá interactuar con el chatbot creado.

4.2.2.1. Fase 2.2.1: Preparación e información del entorno

Antes de empezar a desarrollar el chatbot se recopiló más información sobre Watson. El siguiente paso fue registrarse en *Bluemix* para poder realizar los desarrollos.

Tras el registro en IBM Cloud, se accede a `Crear Servicio` y se selecciona el servicio `Conversation` en la categoría de Watson. Una vez seleccionado el servicio, es importante acceder al apartado `Credenciales del servicio` para obtener los nombres de usuario y contraseña. Con ello se obtendrá el acceso a través de la API.

Después de crearse el servicio, se lanza la herramienta mediante el botón `Launch tool`. Entonces se abrirá una nueva página que permitirá crear nuestro propio *workspace*. Es especialmente relevante anotar el ID del workspace para poder acceder a él desde la API. Este es un contenedor para los artefactos que definen el flujo de la conversación.

Antes de pasar a los prototipos que he desarrollado, es importante identificar y distinguir los elementos que utiliza Watson para el desarrollo del bot: intenciones, entidades y diálogos.

Una *intención* representa el propósito de la entrada del usuario, es decir, las intenciones pueden considerarse como las acciones que los usuarios podrían querer hacer en el asistente.

Para un primer contacto, se definen sólo dos intenciones:

- `#Hola`: para cuando el usuario quiere saludar al chatbot
- `#Adiós`: para cuando el usuario quiera despedirse

Una vez identificados los *intents*, es necesario entrenarlos con las posibles entradas que Watson podría recibir y debería asociar a esa intención. Los ejemplos que se añaden a cada uno, le dicen al servicio que tipo de entrada

Figura 4.3: Intención `#Hola`.

de usuario quiere que coincida con la intención. Cuantos más ejemplos se le proporcionen, más preciso será el servicio para reconocer las intenciones del usuario. Las Figuras 4.3 y 4.4 muestran cómo quedarán estas intenciones en nuestro sistema.

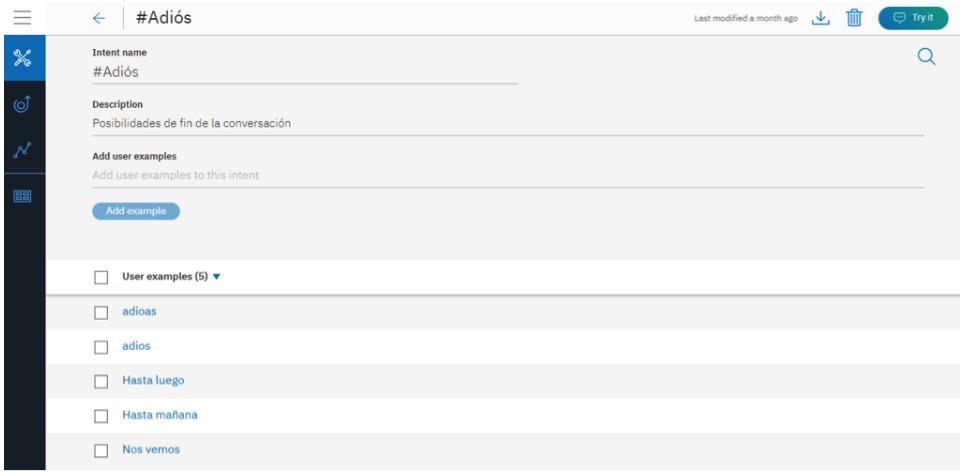


Figura 4.4: Intención `#Adiós`.

Para introducir una mayor complejidad en las conversaciones, se introduce el concepto de *entidad*. La definición de una entidad incluye un conjunto de valores de entidad que pueden usarse como disparadores para diferentes respuestas. El valor de cada una puede tener múltiples sinónimos, definiendo distintas formas para especificar un mismo valor.

Es la forma en que Watson trata de forma significativa las partes de la entrada, para adaptar la respuesta en función del texto especifico que ha introducido el usuario.

Existen además entidades propias del sistema, *system entities* (ver Figura 4.5). Son entidades comunes que se pueden utilizar para cualquier aplicación. Al habilitarlas, es posible obtener rápidamente en el workspace datos que son comunes para muchos casos. La Figura 4.6 presenta un ejemplo de como aparecen en el espacio de trabajo.

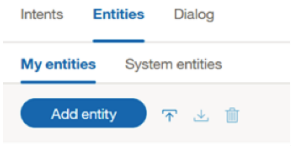


Figura 4.5: Creación de entidades.

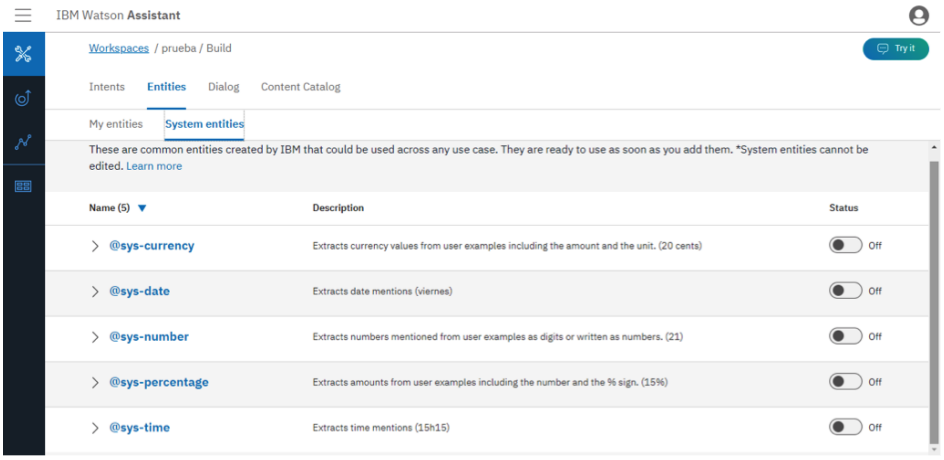


Figura 4.6: Entidades del sistema en el espacio de trabajo.

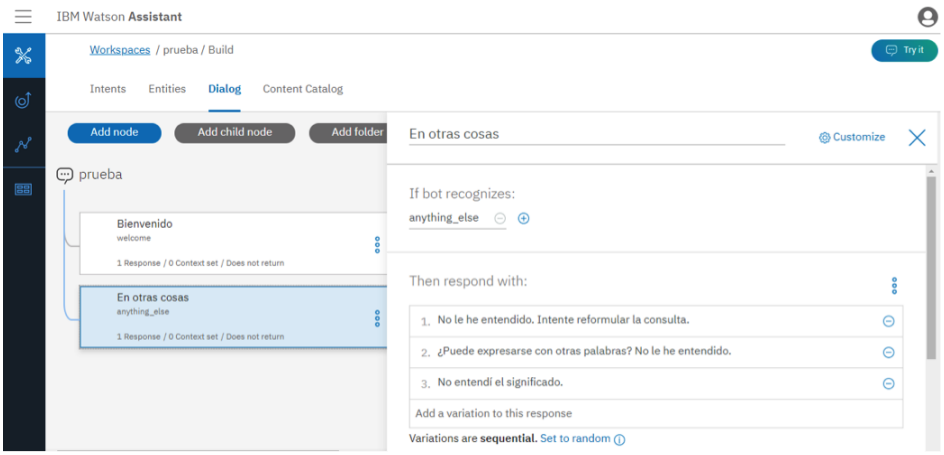


Figura 4.7: Diálogos `Bienvenido` y `En otras cosas`.

Por defecto solo detecta una coincidencia exacta, es decir, cuando la entrada se especifica de una manera en la que coincide exactamente con la definida. Afortunadamente existe también una opción configurable con *fuzzy matching*, que permite al servicio reconocer referencias "similares" a entidades de la entrada del usuario.

Por último, un *dialogo* utiliza las intenciones para identificar el propósito de la entrada —independientemente de la redacción utilizada— y responde de la manera especificada. Define el flujo de la conversación en forma de árbol lógico, donde cada nodo del árbol tiene una condición que lo desencadena basado en la entrada que se le proporcione.

Para un ejemplo simple aparecen por defecto, en el momento de la creación, dos nodos en la ventana de diálogo (ver Figura 4.7):

- `Bienvenido`: Contiene un saludo que se muestra a los usuarios cuando interactúan por primera vez con el bot.

- `En otras cosas`: Contiene frases para responder cuando no reconoce la entrada de los usuarios.

El nodo de diálogo creado se desencadena con la condición de bienvenida. Es un nodo especial e indica que el usuario acaba de comenzar una nueva conversación. Éste especifica al sistema, que cuando una conversación comience, debe responder con uno de los mensajes de bienvenida. Además, permite probar el diálogo para verificar el código y entrenarlo en cualquier momento.

Después de esta introducción sobre el sistema, en la siguiente sección se procede a la creación del prototipo.

4.2.2.2. Fase 2.2.2: Diseño y desarrollo del bot

En esta fase se muestra el proceso seguido para el desarrollo de los dos prototipos. Se identificará la funcionalidad de cada uno, así como el corpus de ejemplos, y el desarrollo de la lógica en IBM Watson.

Los objetivos que se muestran a continuación son los pasos a seguir para el desarrollo de la aplicación final:

- Crear un diálogo complejo, agregando entidades y aclarando el propósito del usuario.

- Implementar el área de trabajo, conectándola a una interfaz de usuario *front-end*, redes sociales o a un canal de mensajería.

Fase 2.2.2a Prototipo Inicial: Bot Turístico de Madrid
Se definen las capacidades de nuestro sistema centrándonos en el asistente turístico sobre Madrid y conociendo los conceptos con los que trabaja Watson.

Descripción de la funcionalidad
En la realización de un buen diseño se deben en cuenta varios aspectos, aunque alguno se aplique para el desarrollo:

1. *Definición del alcance*:
   - ¿Cuál es la funcionalidad de mi bot?
   - ¿Cuáles son las acciones que el bot será capaz de ejecutar?

   Esas preguntas servirán también para *definir las entidades que necesitan usarse*. Por ejemplo, si el cliente pide información del Museo del Prado, ¿le enseño la dirección? ¿le cuento los descuentos? ¿le muestro el horario? ¿le pregunto al usuario que qué le interesa en concreto? ...
2. Identificación de las *intenciones genéricas*: saludos, agradecimientos, ayuda, etc.
3. Identificación de las *intenciones específicas*.
4. Diseño de un *esquema del diálogo*.

Por una parte, se pretende diseñar un asistente virtual sobre el turismo en Madrid. Para obtener un chatbot de cultura y ocio es necesario definir bien el ámbito de las funcionalidades:

- Museos
- Parques y jardines
- Monumentos
- Diversión: cine, teatro, parque de atracciones, zoo

Por otra parte, es hora de definir las acciones que queremos que cubra en estos dominios. La mayoría de estas entidades tienen acciones comunes, como:

- Localización
- Horario
- Precio: online y en taquilla
- Descuentos

Aunque existen otras entidades específicas para cada una, como se muestra a continuación:

- Museos: selección de los cinco mejores museos de arte de Madrid
- Parques y jardines: selección de los cinco mejores parques y jardines de Madrid
- Monumentos: monumentos interesantes de Madrid
- Diversión:
  - Cine: día del espectador
  - Teatro: selección de los cinco mejores teatros de Madrid
  - Parque de atracciones: selección de los parques de atracciones de Madrid
  - Zoo: selección de los zoos de Madrid

A la hora de definir las acciones que deseamos que nuestro bot sea capaz de ejecutar, se podían observar limitaciones al obtener la información. Podemos distinguir como información *estática* aquella que no necesita su consulta en tiempo real, mientras que la información *dinámica* sería la que necesita ser comprobada al momento.

Por ejemplo, respecto del horario de los pases del cine, normalmente el cine siempre retransmite películas en un cierto horario. Pero si queremos conocer el horario para una película en especial, será necesario consultar la cartelera del día concreto para el cine concreto.

Con el teatro pasaría algo parecido, si el usuario busca una obra concreta, será necesario realizar una búsqueda que tendría que acceder a la información actualizada que se utilice.

Por lo que se decidió que el primer prototipo fuera un sistema de preguntas y respuestas, sin que se realizaran búsquedas en ningún sistema externo. Este será un sistema basado en reglas, en el cual las intenciones y entidades sirven para entender las preguntas y dar respuestas. Una intención está definida por un cluster de aceptación, que son expresiones que sirven para preguntar una cuestión. Cuando se pregunta de otra forma se establecen niveles de afinidad con cada uno y, finalmente, se asocia con el que tiene mayor nivel de afinidad.

Desarrollo Lógica IBM

Basándonos en el análisis de la funcionalidad y de las acciones del apartado anterior, se declararán las entidades e intenciones para poder crear un diálogo y, por último, se definirá el diálogo.

Las entidades hacen referencia a los datos que necesitaremos para definir las acciones que queremos que se realicen y ayudar a que el bot sepa exactamente qué hacer. Por ejemplo, se necesita tener respuestas a: ¿dónde está? ¿estará abierto? ¿cuánto cuesta? ¿hay descuentos? ...

Como en este caso la información es estática, hemos seleccionado ejemplos concretos para cada entidad. En la Figura 4.8 se muestran las entidades que se han definido.

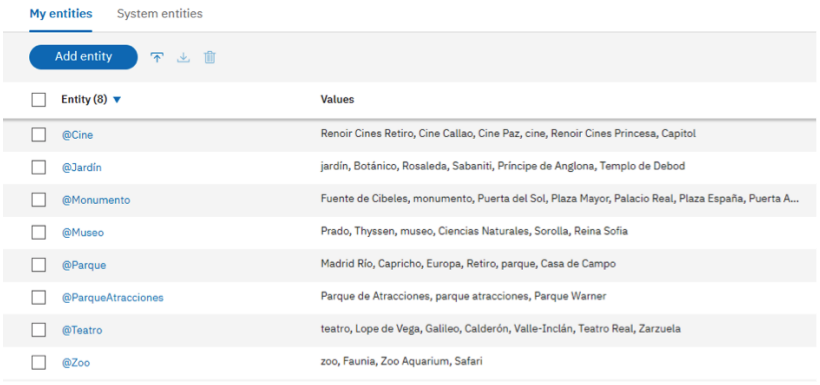


Figura 4.8: Entidades definidas.

A continuación, la Figura 4.9 muestra el caso de la entidad `Museo`, aunque se ha utilizado el mismo esquema para cada una.

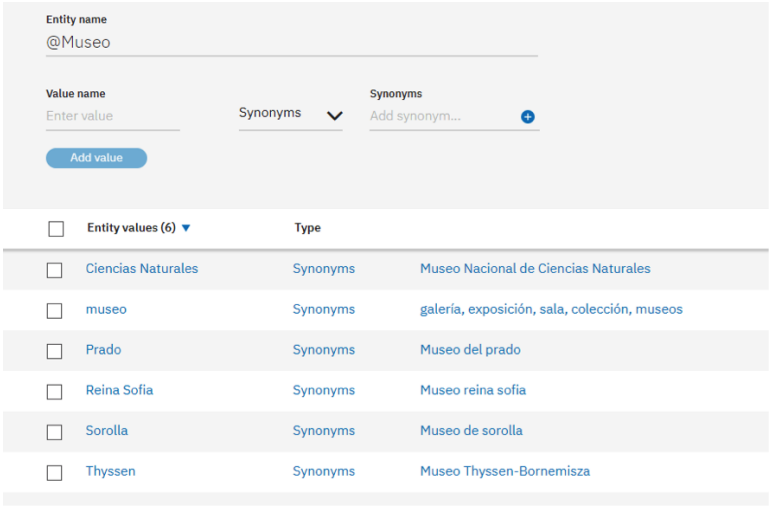


Figura 4.9: Definición de la entidad `Museo`.

Las intenciones hacen referencia a las acciones que puede querer hacer el usuario del bot, por ejemplo, ¿quiero ir al Museo del Prado?. Podemos distinguir entre intenciones genéricas o específicas (ver Figura 4.10).

Intent (11)	Description	Modified	Examples
#Adiós	Posibilidades de fin de la conversación	16 days ago	5
#Ayuda	Información de las funcionalidades del bot	16 days ago	5
#Descuento	Maneras de preguntar por descuentos	16 days ago	5
#DíaEspectador	Maneras de preguntar por el día del espectador	16 days ago	2
#Hola	Posibilidades de inicio de conversación	16 days ago	6
#Horario	Maneras de preguntar por el horario	15 days ago	8
#Localización	Lugar donde se ubica	15 days ago	9
#Precio	Listado de formas de preguntar por la cantidad de la entrada	16 days ago	8
#PrecioOnline	Distintas maneras de preguntar el precio online	16 days ago	4
#PrecioTaquilla	Distintas maneras del precio en taquilla	16 days ago	2
#Selección	Maneras de preguntar por un grupo de sitios	16 days ago	19

Figura 4.10: Definición de intenciones.

Buscamos las distintas maneras en que el usuario podría querer expresar su pregunta. Para el primer caso, podemos diferenciar como intenciones genéricas, a la intención `#Hola` (ver Figura 4.11). La Figura 4.11 también muestra la declaración de la intención de despedida, mientras que la Figura 4.12 muestra la definición de la entidad para la intención de ayuda.

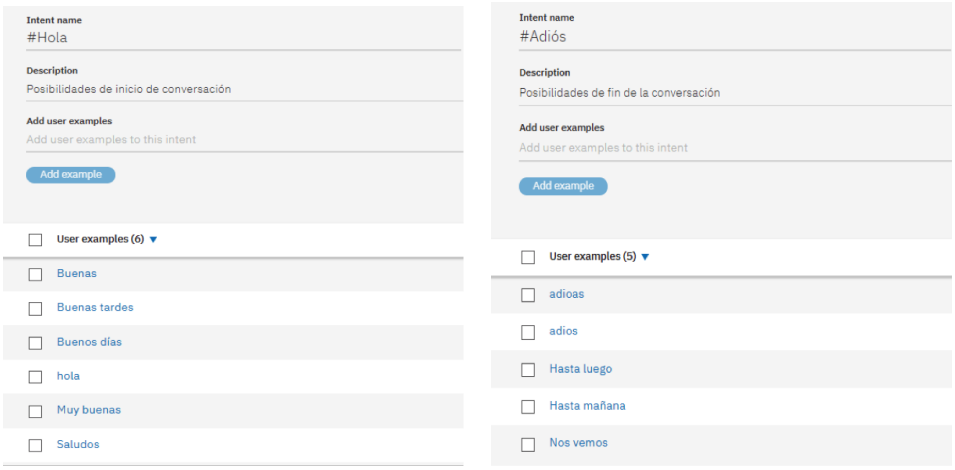


Figura 4.11: Intenciones `#Hola` y `#Adiós`.

Además de estas acciones generales, existen una serie de tareas comunes y otras concretas para cada entidad. Es importante ponerse en el lugar del usuario y pensar como realizaría esa consulta y que frases utilizaría. Una vez pensadas, crearemos nuestras intenciones.

Como acciones comunes entendemos aquellas que comparten todas las entidades y podemos encontrarnos con las siguientes:

- `#Localización`: buscamos como el usuario podría trasmitir la intención de conocer el lugar donde está el museo (ver Figura 4.13).

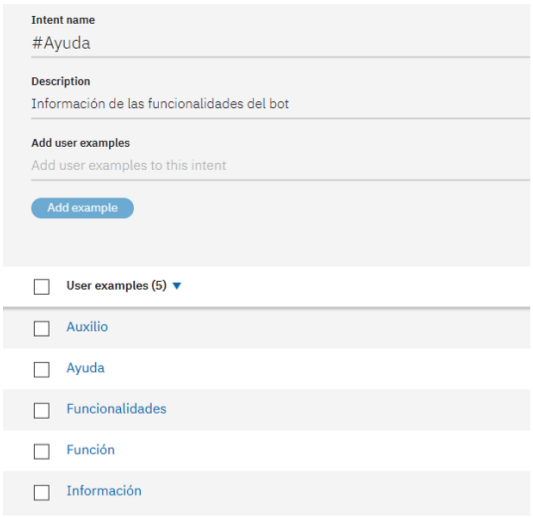


Figura 4.12: Intención `#Ayuda`.

- `#Horario`: listamos las distintas formas en que un usuario podría indicarnos que quiere conocer el horario de un museo de Madrid (ver Figura 4.13).

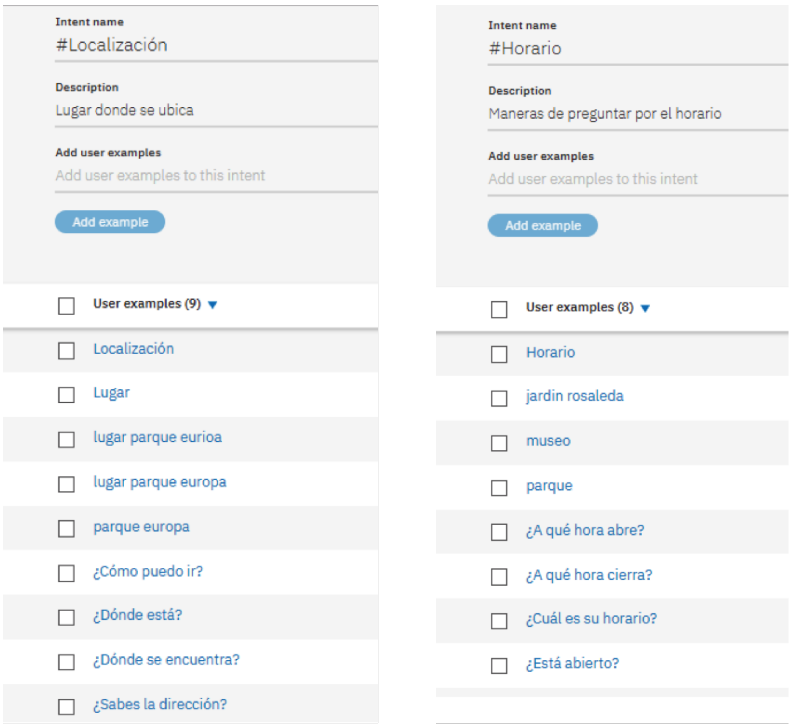


Figura 4.13: Intenciones `#Localización` y `#Horario`.

- Para definir la intención `#Precio`, listamos las distintas maneras en que un usuario podría indicarnos que quiere conocer el precio de un museo de Madrid (ver Figura 4.14).

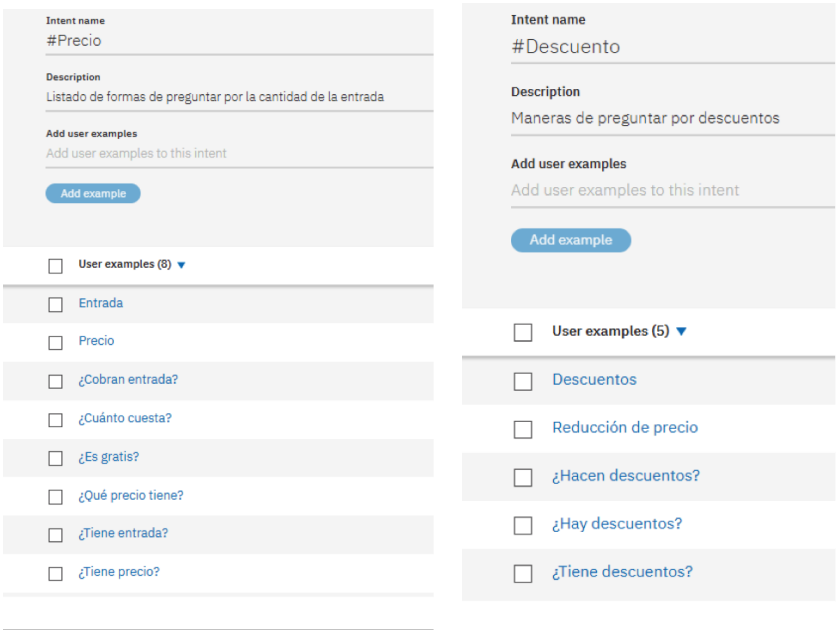


Figura 4.14: Intenciones `#Precio` y `#Descuento`.

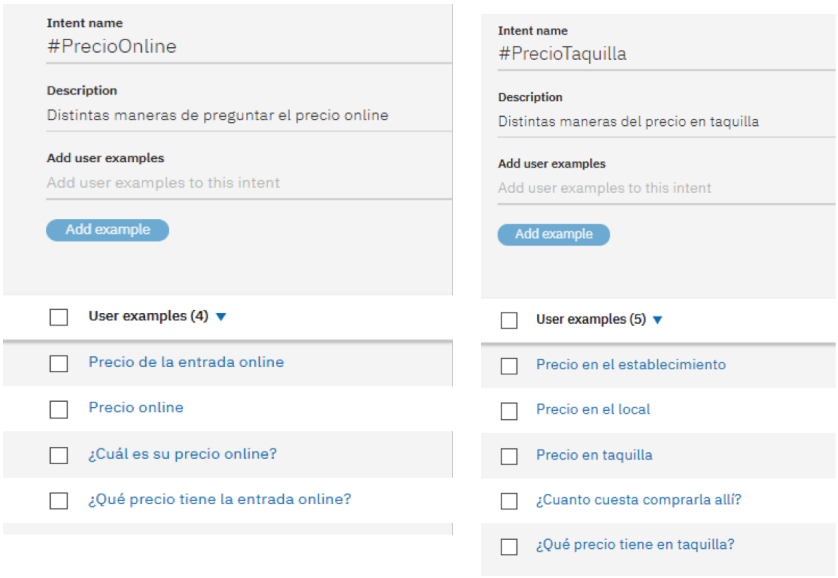


Figura 4.15: Intenciones `#Precio online` y `#Precio en taquilla`.

Además, existen dos opciones para poder realizar la compra normalmente (ver Figura 4.15):

- `#Precio online`
- `#Precio en taquilla`

- `#Descuento`: listamos las distintas maneras en que un usuario podría indicarnos que quiere conocer los descuentos aplicables en el caso de comprar la entrada.

Para las intenciones concretas, definimos en primer lugar la intención `#SeleccionMejoresMuseos` y listamos las distintas formas en que un usuario podría indicarnos que quiere conocer algún museo de Madrid:

1. Selección de los cinco mejores museos de arte de Madrid
2. ¿Me recomiendas algún museo?
3. ¿Conoces algún museo al que pueda ir?
4. ¿Sabes algún museo al que ir?
5. ¿Me dices un museo?
6. Me gustaría visitar un museo
7. Me gustaría ir a un museo
8. Quiero ir a algún museo
9. Museo
10. Arte

Se utiliza el mismo modelo para la selección de los mejores parques y jardines, monumentos, teatros, parques de atracciones y zoológicos.

Otra intención que se ha considerado es conocer el día del espectador. Las distintas maneras que se han seleccionado se muestran en la Figura 4.16.

En este caso, no se contaba con un banco de preguntas reales, de forma que manualmente se han escrito algunas posibilidades. Es importante que queden cubiertas las distintas posibilidades y utilizar muchos sinónimos. Se debe tener en cuenta que con el entrenamiento aumentará su base de conocimiento.

Por último, una vez que se han definido los campos anteriores es importante crear bien el diálogo.

En tercer lugar, definiremos el diálogo y, con ello, las respuestas. El diálogo utiliza las intenciones y las entidades que se identifican con la entrada del

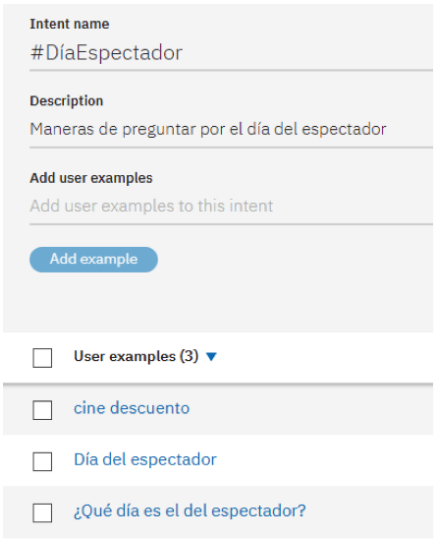


Figura 4.16: Intención `#DíaEspectador`.

usuario más el contexto, para interactuar así con el usuario y proporcionar una respuesta útil.

Para ello necesitamos crear nodos de conversación, que se desencadenarán a partir de la condición preestablecida. Se ha organizado el diálogo en función de las entidades como se muestra en la Figura 4.17.

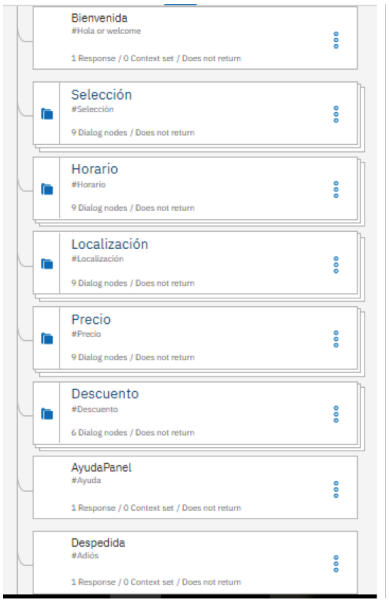


Figura 4.17: Organización del diálogo en función de las entidades.

Dentro de estas carpetas, se enseñará en detalle la creación del contenido

del nodo `horario`. Se sigue el mismo modelo para el resto de intenciones, según se muestra en la Figura 4.18.

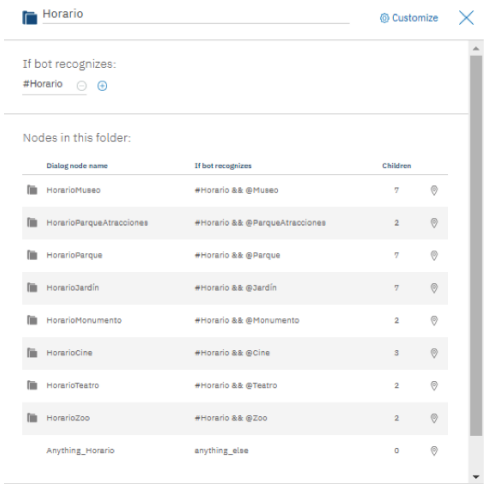


Figura 4.18: Detalle de la creación del contenido del nodo `horario`.

A su vez, cada uno de los nodos carpeta corresponde a las entidades que pueden ser preguntadas por esa intención (ver Figura 4.19). Por ejemplo, para el caso del museo, estaría compuesto de cada uno de los museos que existan.

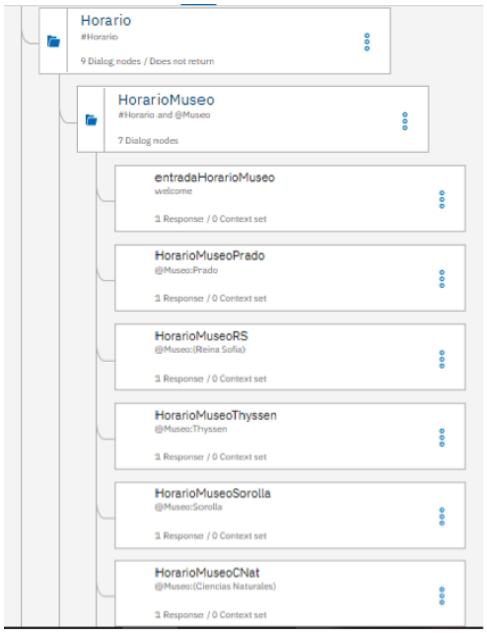


Figura 4.19: Nodo `horario`.

La Figura 4.20 muestra el saludo del bot cuando se inicia el diálogo y la despedida (aleatoria) cuando se cierra el diálogo.

Figura 4.20: Saludo y despedida del bot.

El siguiente proceso es el entrenamiento del chatbot para crear una mayor afinidad entre las respuestas del usuario y los clusters de los que están compuestos cada corriente de la conversación.

Problemas en fase de pruebas

Una vez definido todo lo necesario, lo siguiente fue realizar las pruebas para comprobar su funcionamiento.

Resultó ser ineficiente, pues no daba la sensación de tener una conversación con un humano. Además, la información que responde está explicita en su base de conocimiento y no podría contestar a preguntas con otra información aunque fuera ligeramente distinta.

A pesar de mantener la conversación, como el dominio escogido era tan amplio no aparentaba tener inteligencia. Por ello, se decidió readaptar el proyecto para poder ofrecer al usuario una buena experiencia. En el siguiente apartado se detalla en que consistió la modificación y la manera en que se utiliza el prototipo final.

Fase 2.2.2b Prototipo final: Bot Turístico de Madrid

En esta sección se tratará sobre el desarrollo realizado con IBM Watson, la gestión de la información y la lógica utilizada. Como hemos mencionado en el capítulo anterior, el prototipo final es un recomendador de museos de la comunidad y ciudad Madrid.

Para poder configurar y entrenar el chatbot, accederemos a través de la API de Watson a nuestro bot. La interacción del bot con nuestro orquestador depende de cómo hayamos configurado inicialmente el GUI en la web.

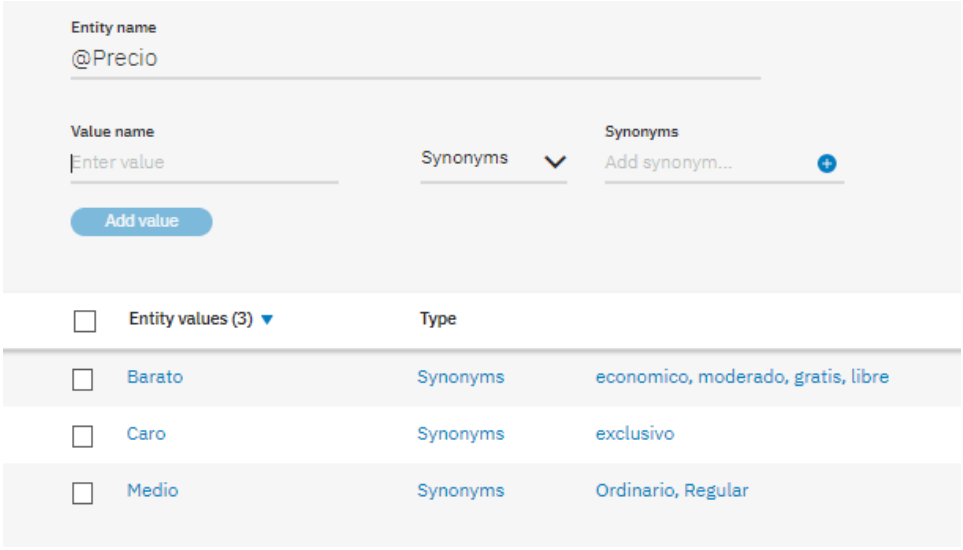


Figura 4.21: Entidad `precio` del museo.

Dicha interfaz, como hemos mencionado en el funcionamiento de la herramienta IBM Watson, permite la definición de entidades. Representan conceptos relacionados con la temática del bot y facilitan su reconocimiento por parte de éste. La plataforma también ofrece la posibilidad de utilizar entidades del sistema como tiempo, números, etc.

Para ayudar al sistema en el proceso de detección de las intenciones del usuario y la clusterización que hará con ella, definimos *intents*. Cada uno de estos *intents* se corresponde con una posible intención del usuario, por ejemplo, saludar.

Definidas las intenciones y entidades, se definen los nodos de diálogo. Sirven para el tratamiento que el servicio realiza con las entidades e intenciones reconocidas y determinan la respuesta.

Una vez configurado el diálogo, se procede al entrenamiento del bot para guiarlo y así facilitar la clusterización que mencionamos previamente.

Ahora pasaremos a detallar las características particulares del recomendador de museos IBM. Para realizar la recomendación, el servicio recogerá las preferencias del usuario en cuanto al precio, localización y tipo de arte que ofrecen los museos.

En cuanto a las entidades definidas, utilizamos una para cada uno de los tres parámetros que el bot utiliza para hacer la recomendación (ver Figuras 4.21, 4.22 y 4.23).

Las intenciones del usuario que pretendemos reconocer son manifestaciones sobre sus preferencias para los parámetros que hemos seleccionado (ver Figuras 4.24 y 4.25).

Figura 4.22: Entidad `lugar` del museo.

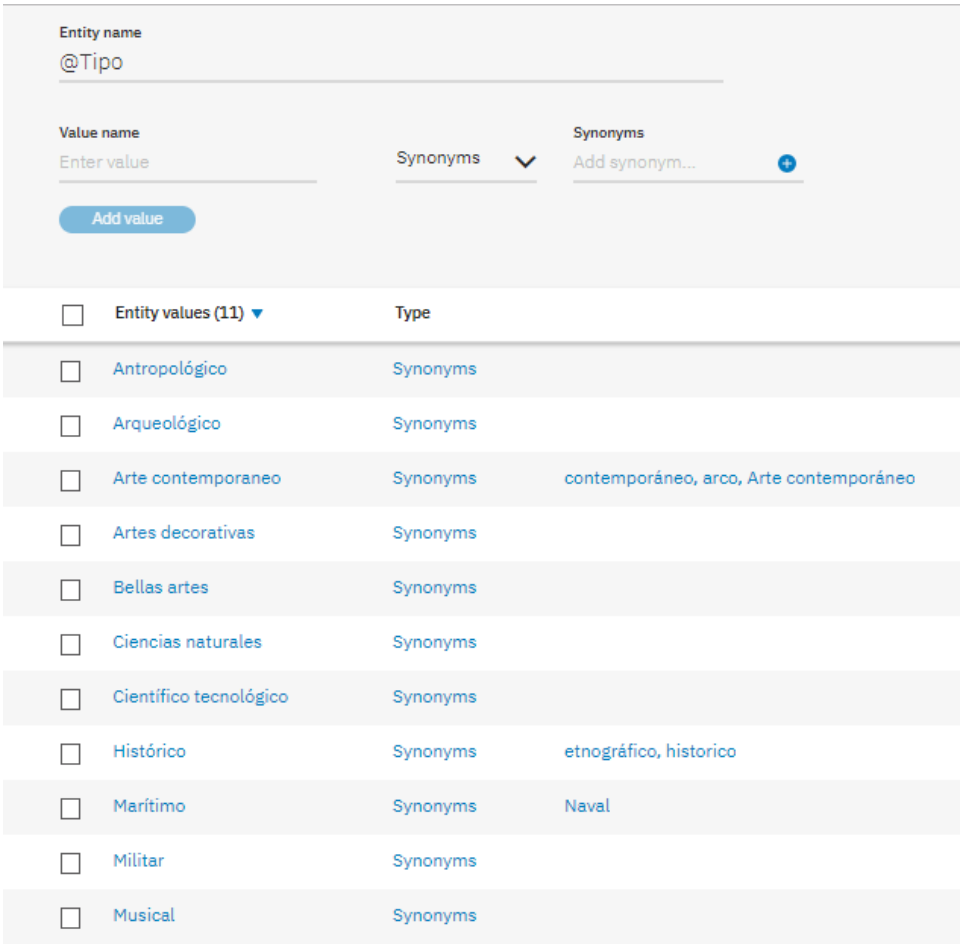


Figura 4.23: Entidad `tipo` del museo.

La estructura del diálogo es plana y los nodos de salida están definidos en las posiciones superiores para que sean los primeros con los que se haga el matching.

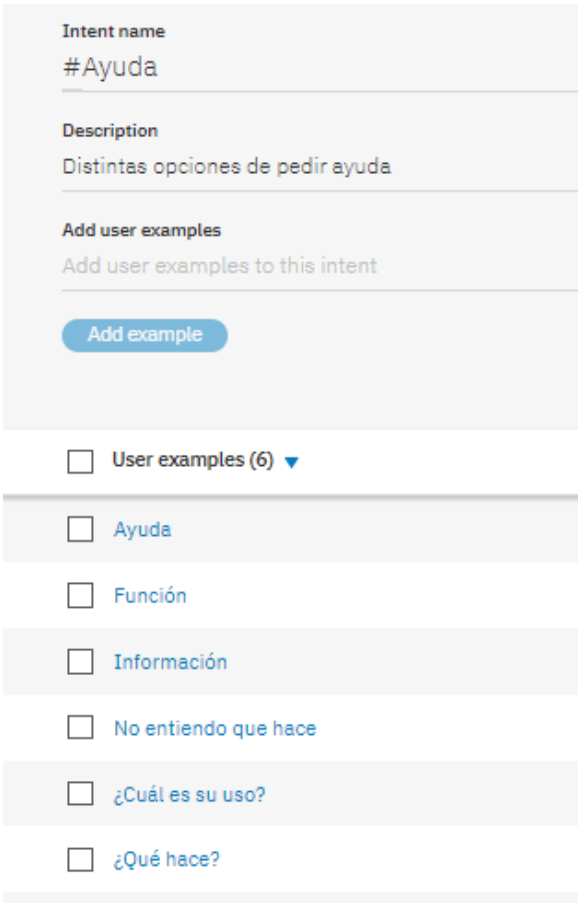


Figura 4.24: Intención `ayuda` en Watson.

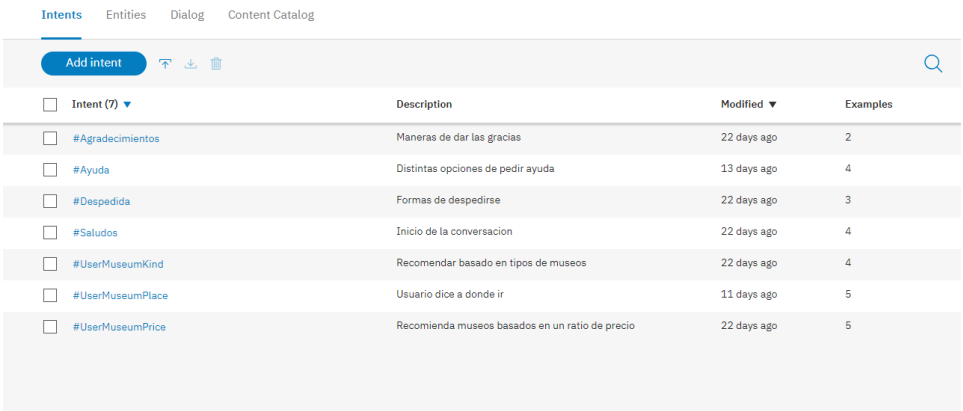


Figura 4.25: Intenciones de Watson.

No necesitamos que sean hijos de otros nodos porque el contexto reconocido por el bot es enviado desde el cliente. Para ello, en el resto de nodos, desde el bot damos valores a determinadas variables que enviamos al orquestador en función de las entidades e intenciones reconocidas (ver Figura 4.26).

Para que la recomendación funcione con efectividad, hemos configurado una base de datos en la que se introducen los museos. Esta ha sido utilizada como fuente de conocimiento.

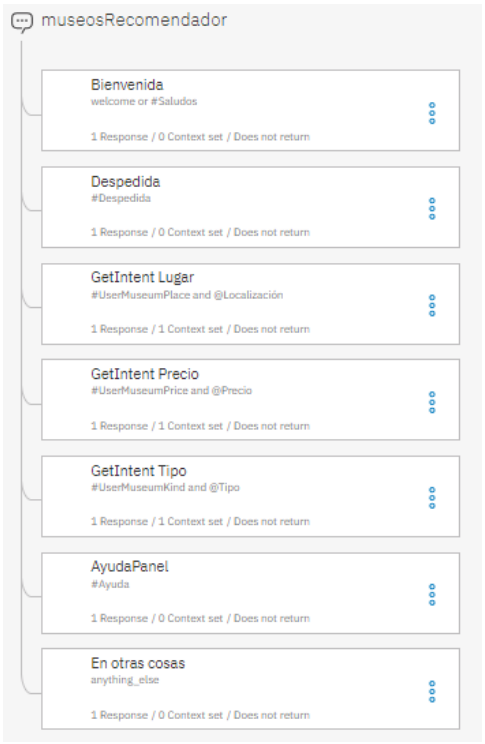


Figura 4.26: Diálogo del recomendador en Watson.

4.3. Fase 3: Instalación y ejecución

Para la instalación de la aplicación se recomienda utilizar el sistema operativo Debian. El siguiente paso será actualizar y descargar los programas necesarios para su ejecución.

Primero, ponemos al día el sistema descargando las últimas actualizaciones con el siguiente comando:

Listado 4.1: Actualizaciones

```
apt-get update && apt-get upgrade -y
```

Después instalamos las siguientes dependencias:

Listado 4.2: Dependencias

```
# Descarga MongoDB
apt-get install mongodb

# Descarga Python3
apt-get install python3

# Descarga pip3
apt-get install python3-pip

# Descarga Watson
pip install --upgrade Watson-developer-cloud

# Descarga PyMongo
pip install --upgrade pymongo
```

Una vez que están funcionando estos programas y librerías, el siguiente paso será montar la base de datos. Para iniciar y detener los servicios de mongoDB se deberá escribir:

Listado 4.3: Iniciar/Detener MongoDB

```
# Iniciar MongoDB
systemctl start mongodb

# Detener MongoDB
systemctl stop mongodb
```

Después de completar la instalación, es hora de crear la colección en Mongo:

Listado 4.4: Creación

```
# Lanzar la consola de mongoDB
mongo

# Crear una base de datos
use myNewDb
# Insertar los elementos del documento: Anexo
   Datos Museos Final
db.myNewCollection.insertMany( {} )
# Comprobar que se han cargado correctamente
db.myNewCollection.find( {} )
```

Con este último comando se obtiene un resultado similar a este código:

Listado 4.5: Actualizaciones

```
db.myCollect.find({},{_id:0}).pretty()
{
    "nombre" : "Reina Sofía",
    "lugar" : "Centro",
    "precio" : "Medio",
    "tipo" : "Bellas artes"
}
{
    "nombre" : "CA2M",
    "lugar" : "Centro",
    "precio" : "Barato",
    "tipo" : "Arte contemporáneo"
}
{
    "nombre" : "Museo Nacional de ciencia y
       tecnología",
    "lugar" : "Sur",
    "precio" : "Caro",
    "tipo" : "Científico Tecnológico"
}
```

Para finalizar se debe ejecutar el código de la aplicación en otra terminal y así crear las conexiones del bot con Telegram.

Capítulo 5

Conclusión

En este capítulo se presentan las conclusiones de este proyecto y se analizan las posibles mejoras en un trabajo futuro.

5.1. Conclusiones

En este proyecto se ha realizado una investigación inicial de las distintas técnicas que existen para el desarrollo de un chatbot. Además se ha creado un asistente que recomienda museos de la comunidad y ciudad de Madrid en base a unas características que se reciben por parte del usuario.

La arquitectura final permite ser exportada e instalada en otros sistemas, debido al diseño elegido. Se pueden tratar las intenciones de los clientes gracias a la capacidad y facilidad de IBM Watson para reconocer el lenguaje natural. En el diseño, también se ha tratado de proporcionar al usuario una buena experiencia a través de una interfaz agradable y de fácil uso, que se consigue con Telegram.

Ha sido interesante enfrentarme desde cero a un proyecto completo: desarrollo, pruebas y documentación. Tanto las tecnologías que he utilizado como los lenguajes que se requerían fueron novedosos para mí. Superados esos obstáculos iniciales aparecieron muchos otros, como los problemas de licencias, compatibilidad de versiones, etc., pero finalmente se ha conseguido el reto de desarrollar este chatbot y completar la presente memoria.

La principal aportación de este trabajo es servir de referencia respecto a las diferentes tecnologías actuales para el desarrollo de chatbots y respecto al diseño de la arquitectura a emplear en otros proyectos análogos.

Con este Trabajo de Fin de Grado, se reflejan las capacidades y conocimientos adquiridos en las distintas asignaturas a lo largo de toda la carrera.

5.2. Trabajos futuros

En lo referente a las funcionalidades, algunas de las posibles mejoras para este proyecto y que sería interesante introducir son:

- *Text-to-speech*: Añadir voz al chatbot, es decir, que pueda reproducirse en audio la respuesta que facilita el bot.
- *Speech-to-text*: Dar la posibilidad al usuario de comunicarse por voz. Introducir el reconocimiento de voz supondría una gran mejora para una mejor interacción con el usuario.

Para aportar una mejor experiencia al usuario y crear una relación de confianza, sería interesante aportar personalizaciones y saber si el cliente es nuevo o ya conoce nuestro servicio. En este último caso, debería recomendarle museos distintos de los que ya hubiera visitado.

Respecto de los contenidos del chatbot, se podría ampliar el campo de los temas dentro de la conversación.

Un trabajo futuro podría ser también integrar este chatbot en otras plataformas y canales distintos del empleado.

En resumen, las mejoras comentadas anteriormente serían interesantes para implementarse en un nuevo prototipo en un futuro.

Chapter 5

Conclusions and Future Work

This chapter presents the conclusions of this project and analyzes possible improvements in future work.

5.1. Conclusions

In this project an initial investigation of the different techniques that exist for the development of a chatbot has been carried out. In addition, we created an assistant that recommends museums in the community and city of Madrid based on characteristics that are received by the user.

The final architecture could be exported and installed in other systems, due to the chosen design. The intentions of the clients can be treated thanks to the ability and facility of IBM Watson to recognize natural language. In the design, we have also tried to provide the user with a good experience through a user-friendly interface, which is achieved with Telegram.

It has been interesting to face a complete project from scratch: development, testing and documentation. Both the technologies that I used and the languages that were required were novel for me. Once these initial obstacles were overcome, many others appeared, such as license problems, version compatibility, etc., but finally the challenge of developing this chatbot and completing this memory has been achieved.

The main contribution of this work is to serve as a reference regarding the different current technologies for the development of chatbots and regarding the design of the architecture to be used in other analogous projects.

This Final Degree Project reflects the skills and knowledge acquired in the different subjects studied throughout the entire degree.

5.2. Future Work

Regarding the functionalities, some of the possible improvements for this project and that would be interesting to introduce are:

- *Text-to-speech*: Add voice to the chatbot, that is, that the response provided by the bot can be reproduced in audio.
- *Speech-to-text*: Give the user the possibility to communicate by voice. Introducing voice recognition would be a great improvement for better interaction with the user.

To provide a better user experience and create a relationship of trust, it would be interesting to provide personalizations and know if the customer is new or already knows our service. In the latter case, it should recommend museums other than those he had already visited.

Regarding the content of the chatbot, the field of topics within the conversation could be expanded.

A future job could also be to integrate this chatbot into other platforms and channels other than the one that has been used.

In summary, it would be interesting to implement the improvements mentioned above in a new prototype in the future.

Apéndice A

Código en *Python*

A.1. recomendadorMR.py

Este código hace referencia al recomendador de museos implementado en Python para ser utilizado a través de la consola.

Listado A.1: recomendadorMR.py

```
import watson_developer_cloud

# Set up Assistant service.
service = watson_developer_cloud.AssistantV1(
  username = 'ca0de99b-87b8-4000-9279-9
     e74a4c1ce51',
  password = '50tbeLtlY8dN',
  version = '2018-04-22'
)
workspace_id = '2bcd2bc7-ffcb-4d0b-9e31-
   e48295abfda1'

# Initialize with empty value to start the
   conversation.
user_input = ''
context = {}
current_action = ''

targetT = ''
targetL = ''
targetP = ''

# Main input/output loop
while current_action != 'end_conversation':
```

```
# Send message to Assistant service.
response = service.message(
  workspace_id = workspace_id,
  input = {
    'text': user_input
  },
  context = {
    '_node_output_map' : {
       'targetTipoMuseo': targetT,
       'targetLugarMuseo': targetL,
       'targetPrecioMuseo': targetP}
  },
)

# Print the output from dialog, if any.
if response['output']['text']:
  print(response['output']['text'][0])

# Update the stored context with the latest
   received from the dialog.
context = response['context']

#response
if 'targetLugarMuseo' in response['context']:
  targetL = response['context']['
     targetLugarMuseo']
if 'targetPrecioMuseo' in response['context']:
  targetP = response['context']['
     targetPrecioMuseo']
if 'targetTipoMuseo' in response['context']:
  targetT = response['context']['
     targetTipoMuseo']

# Check for action flags sent by the dialog.
if 'action' in response['output']:
  current_action = response['output']['action']

# Han sido registradas las tres variables
# Se puede realizar la consulta en la bbdd
if targetL != '' and targetP != '' and targetT
   != '' and  current_action != '
   end_conversation':
  print('BUSCAR RECOMENDACION')
```

```
  # If we're not done, prompt for next round of
     input.
  if current_action != 'end_conversation':
    user_input = input('>> ')
```

A.2. botWatson.py

Este código se utiliza para poner en marcha el servicio del recomendador de museos implementado en Python comunicándose con IBM-Watson y conectado a una base de datos MongoDB. El usuario final puede acceder al recomendador a través de la aplicación de Telegram.

Listado A.2: botWatson.py

```
import requests
import json
import watson_developer_cloud
from pymongo import MongoClient

# Set up Assistant service.
service = watson_developer_cloud.AssistantV1(
  username = 'ca0de99b-87b8-4000-9279-9
     e74a4c1ce51',
  password = '50tbeLtlY8dN',
  version = '2018-04-22'
)
workspace_id = '2bcd2bc7-ffcb-4d0b-9e31-
   e48295abfda1'

# ArtMadridBot
BOT_TOKEN = '576567955:AAGOxHTlHbp0-uJExQSmJ-Fdv-
   piOYVMWoE'
LONG_POLLING_TIMEOUT = 20
REQUEST_TIMEOUT = LONG_POLLING_TIMEOUT * 2

quit=False
salidaUser = ''


def on_receive_message(event):
   # Def variables
    targetT = ''
    targetL = ''
    targetP = ''

    message_from = event['from']['first_name']
    user_input = event['text']
    print("[", message_from,"] << ", user_input,
       sep="")
```

```
    # Send message to Assistant service.
    response = service.message(
        workspace_id = workspace_id,
        input = {
            'text': user_input
        },
    )

    # Save the variable if it is saying.
    if 'targetLugarMuseo' in response['context']:
        targetL = response['context']['
           targetLugarMuseo']
    if 'targetPrecioMuseo' in response['context'
       ]:
        targetP = response['context']['
           targetPrecioMuseo']
    if 'targetTipoMuseo' in response['context']:
        targetT = response['context']['
           targetTipoMuseo']

    # Check the variables and query in database.
    if compruebaAtributos(event, targetL, targetP
       , targetT):
        consultaBD(event,targetL,targetP,targetT)
           ;
    # If not
    else:
        if response['output']['text']:
            salidaUser = response['output']['text
               '][0]
            send_message(event['from']['id'],
               salidaUser, False, event['
               message_id'])

def on_receive_callbackQuery(event):
    message_from = event['from']['first_name'];
    print("[", message_from,"] [CallbackQuery] ",
        event["data"], sep="")
    if event["data"] == "editthismessage":
        api_request('editMessageText', {
            "chat_id": event['message']['chat']['
               id'],
            "message_id": event['message']['
               message_id'],
```

```
            "text": 'Message edited! :)',
        })
    else:
        api_request('answerCallbackQuery', {
            "callback_query_id": event['id'],
            "text": 'Patience you must have my
               young padawan',
            #"show_alert" => True,
        })

def consultaBD(event,targetL,targetP,targetT):

    # Creamos una bd: prueba
    client = MongoClient()
    db = client.myDatabase
    #Creamos una coleccion: myCollect
    myCollect = db.myCollect

    # find()query
    if targetL != '':
        cursor = myCollect.find( {"lugar":targetL
           } )
        for mus in cursor:
            send_message(event['from']['id'], 'Mi
                recomendacion es que vayas a : '+
               mus["nombre"]+' Espero que te guste
               !', False, event['message_id'])

    # find()query
    if targetP != '':
        cursor = myCollect.find( {"precio":
           targetP} )
        for mus in cursor:
            send_message(event['from']['id'], 'Mi
                recomendacion es que vayas a : '+
               mus["nombre"]+' Espero que te guste
               !', False, event['message_id'])

    # find()query
    if targetT != '':
        cursor = myCollect.find( {"tipo":targetT}
            )
        for mus in cursor:
            send_message(event['from']['id'], 'Mi
```

```
                recomendacion es que vayas a : '+
               mus["nombre"]+' Espero que te guste
               !', False, event['message_id'])

    # Close conexion
    client.close()

def compruebaAtributos(event,targetL,targetP,
   targetT):
    if targetL != '':
        return True
    if targetP != '':
        return True
    if targetT != '':
        return True
    return False

def get_bot_username():
    return api_request('getMe')["username"]

def run():
    last_message_update_id = 0
    while not quit:
        response = api_request('getUpdates', {
            "offset": last_message_update_id + 1,
            "limit": 100,
            "timeout": LONG_POLLING_TIMEOUT
        })
        for update in response:
            if update["update_id"] >
               last_message_update_id:
                last_message_update_id = update["
                   update_id"]
            run_event(update)

def send_message(chat_id, text,
   disable_web_page_preview=False,
   reply_to_message_id=None, reply_markup=None):
    if reply_markup != None:
        reply_markup = json.dumps(reply_markup)
    response = api_request('sendMessage', {
        "chat_id": chat_id,
        "text": text,
        "disable_web_page_preview":
```

```
            disable_web_page_preview,
        "reply_to_message_id":
            reply_to_message_id,
        "reply_markup": reply_markup,
        "parse_mode": 'HTML',
    })
    run_event(response)

def api_request(method, parameters={}, files=None
    ):
    url = "https://api.telegram.org/bot{token}/{
        method}".format(token=BOT_TOKEN, method=
        method)

    http_method = 'get'

    try:
        response = requests.request(http_method,
            url, timeout=REQUEST_TIMEOUT, params=
            parameters, files=files)
    except requests.RequestException as e:
        raise Exception(str(e))

    try:
        result = response.json()
    except ValueError:  # typo on the url, no
        json to decode
        raise Exception('Error: Invalid URL',
            response.status_code)

    if not (response.status_code is requests.
        codes.ok):
        raise Exception(result['description'],
            response.status_code)  # Server
            reported error

    if not result["ok"]:
        raise Exception("Telegram API sent a non
            OK response")  # Telegram API reported
            error

    return result["result"]

def run_event(event):
```

```
    if "message" in event:
        run_message_event(event["message"])
    elif "callback_query" in event:
        on_receive_callbackQuery(event["
           callback_query"])

def run_message_event(event):
    if "text" in event:
        on_receive_message(event)
    if "new_chat_participant" in event:
        #on_new_chat_participant(event)
        print("new_chat_participant")
    if "left_chat_participant" in event:
        #on_left_chat_participant(event)
        print("left_chat_participant")
    if "audio" in event:
        #on_receive_audio(event)
        print("audio")
    if "document" in event:
        #on_receive_document(event)
        print("document")
    if "photo" in event:
        #on_receive_photo(event)
        print("photo")
    if "sticker" in event:
        #on_receive_sticker(event)
        print("sticker")
    if "video" in event:
        #on_receive_video(event)
        print("video")
    if "contact" in event:
        #on_receive_contact(event)
        print("contact")
    if "location" in event:
        #on_receive_location(event)
        print("location")
    if "new_chat_title" in event:
        #on_new_chat_title(event)
        print("new_chat_title")
    if "new_chat_photo" in event:
        #on_new_chat_photo(event)
        print("new_chat_photo")
    if "delete_chat_photo" in event:
        #on_delete_chat_photo(event)
```

```
        print("delete_chat_photo")
    if "group_chat_created" in event:
        #on_group_chat_created(event)
        print("group_chat_created")

run()
```

Bibliografía

Venceréis, pero no convenceréis. Venceréis porque tenéis sobrada fuerza bruta; pero no convenceréis, porque convencer significa persuadir. Y para persuadir necesitáis algo que os falta: razón y derecho en la lucha.

Miguel de Unamuno y Jugo

Akiwatkar, R. What are the best API's and frameworks to build your own chatbot?. Quora. `http://www.quora.com/What-are-the-best-APIs-and-frameworks-to-build-your-own-chatbot/answer/Rohit-Akiwatkar`, 2017. [Online; fecha de acceso: 14-Mayo-2018].

Amazon. Introducción a Amazon Lex. `http://aws.amazon.com/es/lex`, 2018. [Online; fecha de acceso: 14-Mayo-2018].

Amazon Alexa Web Portal. Amazon Alexa. `http://developer.amazon.com/alexa`, 2017. [Online; fecha de acceso: 14-Mayo-2018].

Ananthavel. Best chatbot platforms to build a chatbot. Medium. `http://chatbotslife.com/13b846a59ddd`, 2018. [Online; fecha de acceso: 14-Mayo-2018].

Aristimuño, L. Los mejores bots de juegos para Telegram, Profesional Review. `http://www.profesionalreview.com/2017/05/11/los-mejores-bots-juegos-telegram`, 2017. [Online; fecha de acceso: 14-Mayo-2018].

Banker, K. *MongoDB in action*. Manning Publications Co., 2011.

Briz, V. Las 3 mejores plataformas para tu chatbot. Medium. `http://code.likeagirl.io/4b14e83428b9`, 2017. [Online; fecha de acceso: 14-Mayo-2018].

Business Insider. Google's artificial-intelligence bot says the purpose of living is "to live forever". `http://www.businessinsider.com/`

google-tests-new-artificial-intelligence-chatbot-2015-6, 2015. [Online; fecha de acceso: 14-Mayo-2018].

Business Insider. Amazon joins the chatbot conversation. http://www.businessinsider.com/amazon-joins-the-chatbot-conversation-2016-12, 2016. [Online; fecha de acceso: 14-Mayo-2018].

Cannon, B. Python developer's guide documentation. *Read the Docs*, 2018.

Carbonell, J. El procesamiento del lenguaje natural, tecnología en transición. En *Actas del Congreso de la Lengua Española: Sevilla, 7 al 10 octubre, 1992*, páginas 247–250. Instituto Cervantes, 1994.

CCM. Correo no deseado (spam). http://es.ccm.net/contents/39-correo-no-deseado-spam, 2013. [Online; fecha de acceso: 14-Mayo-2018].

Cerdas Mendez, D. Historia de la Inteligencia Artificial relacionada con los chatbots. Medium. http://planetachatbot.com/41a6cda22906, 2016. [Online; fecha de acceso: 14-Mayo-2018].

Chakraborty, P. An introduction to chatbots. Medium. http://whatarechatbots.com/34a6a123796a, 2017. [Online; fecha de acceso: 14-Mayo-2018].

Chodorow, K. *MongoDB: The Definitive Guide: Powerful and Scalable Data Storage*. O'Reilly Media, Inc., 2013.

Claire by 30Secondstofly. The ultimate travel bot list. http://www.30secondstofly.com/ai-software/ultimate-travel-bot-list, 2016. [Online; fecha de acceso: 14-Mayo-2018].

Condé Nast Traveler. La revolución de los bots cambiará tu forma de viajar. http://www.traveler.es/viajes-urbanos/articulos/la-revolucion-de-los-bots-cambiara-tu-forma-de-viajar/8995, 2016. [Online; fecha de acceso: 14-Mayo-2018].

Consejo General de Colegios Oficiales de Farmacéuticos. Qué es *Bot PLUS App*. http://www.portalfarma.com/inicio/botplus20/botplus20app, 2018. [Online; fecha de acceso: 14-Mayo-2018].

Davydova, O. 25 chatbot platforms: A comparative table. http://chatbotsjournal.com/aeefc932eaff, 2017. [Online; fecha de acceso: 14-Mayo-2018].

Dialogflow. Integration of Dialogflow with Slack. http://dialogflow.com/docs/integrations/slack, 2018. [Online; fecha de acceso: 14-Mayo-2018].

Dunham, K. y Melnick, J. *Malicious bots: An inside look into the cybercriminal underground of the Internet*. CrC Press, 2008.

Ebling, M. R. Can cognitive assistants disappear? *IEEE Pervasive Computing*, vol. 15(3), páginas 4–6, 2016.

Forlizzi, J. y DiSalvo, C. Service robots in the domestic environment: a study of the Roomba vacuum in the home. En *Proceedings of the 1st ACM SIGCHI/SIGART Conference on Human-robot interaction*, páginas 258–265. ACM, 2006.

Fujita, M. AIBO: Toward the era of digital creatures. *The International Journal of Robotics Research*, vol. 20(10), páginas 781–794, 2001.

Futurizable. Estado del arte en el desarrollo de chatbots a nivel mundial. `http://futurizable.com/chatbot`, 2017. [Online; fecha de acceso: 14-Mayo-2018].

García Moreno, C. ¿Qué es el Deep Learning y para qué sirve? `http://www.indracompany.com/es/blogneo/deep-learning-sirve`, 2018. [Online; fecha de acceso: 14-Mayo-2018].

Guttag, J. V. *Introduction to computation and programming using Python*. Mit Press, 2013.

High, R. The era of cognitive systems: An inside look at IBM Watson and how it works. *IBM Corporation, Redbooks*, 2012.

IBM. Watson. `http://www.ibm.com/watson/how-to-build-a-chatbot`, 2018a. [Online; fecha de acceso: 14-Mayo-2018].

IBM. Watson assistant. `http://www.ibm.com/watson/services/conversation`, 2018b. [Online; fecha de acceso: 14-Mayo-2018].

Inbenta. Historia del motor de búsqueda: de las fichas a la IA del chatbot. `http://www.inbenta.com/mx/blog/historia-del-motor-de-busqueda-de-las-fichas-la-ia-del-chatbot`, 2017. [Online; fecha de acceso: 14-Mayo-2018].

Kaspersky Lab. ¿Qué es un botnet? `http://www.kaspersky.es/blog/que-es-un-botnet`, 2013. [Online; fecha de acceso: 14-Mayo-2018].

Knuth, D. E. *The TEX book*. Addison-Wesley Professional., 1986.

Kuhlman, D. *A Python book: Beginning Python, advanced Python, and Python exercises*. Dave Kuhlman, 2009.

Lamport, L. *LATEX: A Document Preparation System, 2nd Edition*. Addison-Wesley Professional, 1994.

Malizia, N. Understanding Telegram inline bots. http://unnikked.ga/73ac9aeea643, 2016. [Online; fecha de acceso: 14-Mayo-2018].

Marupaka, P. Chatbots (of) the future!. The Startup. http://medium.com/swlh/86b5bf762bb4, 2018. [Online; fecha de acceso: 14-Mayo-2018].

Mas Digital. 10 errores graves que se comenten con los chatbots. Medium. https://www.masdigital.net/nuestro-blog/10-errores-graves-que-se-comenten-con-los-chatbots, 2017. [Online; fecha de acceso: 14-Mayo-2018].

Mauldin, M. L. ChatterBots, TinyMUDs, and the Turing test: Entering the Loebner prize competition. En *AAAI*, vol. 94, páginas 16–21. 1994.

McCarthy, J. *Programs with common sense*. RLE and MIT Computation Center, 1960.

Membrey, P., Plugge, E. y Hawkins, D. *The definitive guide to MongoDB: the noSQL database for cloud and desktop computing*. Apress, 2011.

Microsoft. Bot Service documentation. http://docs.microsoft.com/en-us/azure/bot-service, 2018. [Online; fecha de acceso: 14-Mayo-2018].

Microsoft Cortana. Cortana: Tu asistente virtual y personal inteligente, Microsoft. http://www.microsoft.com/es-es/windows/cortana, 2017. [Online; fecha de acceso: 14-Mayo-2018].

Moreno, A. Aplicaciones del procesamiento del lenguaje natural. Instituto de Ingeniería del Conocimiento. http://www.iic.uam.es/inteligencia/aplicaciones-procesamiento-lenguaje-natural, 2017. [Online; fecha de acceso: 14-Mayo-2018].

Ng, A., Ngiam, J., Foo, C. Y., Mai, Y. y Suen, C. UFLDL Tutorial. Stanford University. http://ufldl.stanford.edu/wiki/index.php/UFLDL_Tutorial, 2013. [Online; fecha de acceso: 14-Mayo-2018].

Nilsson, N. J. *Principles of Artificial Intelligence*. Morgan Kaufmann, 2014.

Norvig, P. *Paradigms of Artificial Intelligence programming*. Morgan Kaufmann, 1992.

Python Software Foundation. Python 3.6 release. 2017.

Rodríguez, T. Machine Learning y Deep learning: cómo entender las claves del presente y futuro de la inteligencia artificial. Xataka. http://www.xataka.com/robotica-e-ia/

machine-learning-y-deep-learning-como-entender-las-claves-del-presente-y-futuro-de-la-
2017. [Online; fecha de acceso: 14-Mayo-2018].

Russell, S. J. y Norvig, P. *Artificial Intelligence: A modern approach*. Pearson, 2016.

Salandra, G. China, Wechat, and the origins of chatbots, Chatbots Magazine. http://chatbotsmagazine.com/89c481f15a44, 2017. [Online; fecha de acceso: 14-Mayo-2018].

Saygin, A. P., Cicekli, I. y Akman, V. Turing test: 50 years later. *Minds and machines*, vol. 10(4), páginas 463–518, 2000.

Schlicht, M. The complete beginner's guide to chatbots. Medium. http://chatbotsmagazine.com/8280b7b906ca, 2016. [Online; fecha de acceso: 14-Mayo-2018].

Sierra, M. Los chatbots tienen un problema: la mala ortografía de los milenials. Vozpopuli. https://www.vozpopuli.com/altavoz/tecnologia/chatbot-problema-ortografia-jovenes-inteligencia-artificial-milenials_0_1044796061.html, 2017. [Online; fecha de acceso: 14-Mayo-2018].

Singh, T. What is the best way to learn and write an AI chatbot?. Quora. http://www.quora.com/What-is-the-best-way-to-learn-and-write-an-AI-Chatbot-What-are-the-latest-research-pap
2017. [Online; fecha de acceso: 14-Mayo-2018].

Siri, Apple. Official website. http://www.apple.com/ios/siri, 2014. [Online; fecha de acceso: 14-Mayo-2018].

TayTweets. The official account of Tay. http://twitter.com/tayandyou, 2017. [Online; fecha de acceso: 14-Mayo-2018].

Telegram. Telegram web portal. http://telegram.org, 2018. [Online; fecha de acceso: 14-Mayo-2018].

Telegram group. Python Telegram Bot. Github. http://github.com/python-telegram-bot/python-telegram-bot, 2013. [Online; fecha de acceso: 14-Mayo-2018].

Turing, A. M. Computing machinery and intelligence. *Mind*, vol. 59(236), páginas 433–460, 1950.

Twitter. El Bot Informativo. http://twitter.com/bot_informativo, 2018. [Online; fecha de acceso: 14-Mayo-2018].

Van Rossum, G. y Drake, F. L. *The Python language reference manual*. Network Theory Ltd., 2011.

Vinyals, O. y Le, Q. A neural conversational model. *arXiv preprint arXiv:1506.05869*, 2015.

Wallace, R. S. The anatomy of ALICE. *Parsing the Turing Test*, páginas 181–210, 2009.

Weizenbaum, J. ELIZA—a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, vol. 9(1), páginas 36–45, 1966.

Wikipedia. Internet bot — Wikipedia, the free encyclopedia. `http://es.wikipedia.org/wiki/Bot`, 2013. [Online; fecha de acceso: 14-Mayo-2018].

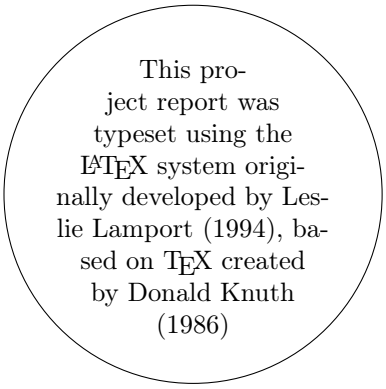
Wikipedia. Telegram Bot API — Wikipedia, the free encyclopedia. `https://es.wikipedia.org/wiki/Telegram_Bot_API`, 2016. [Online; fecha de acceso: 14-Mayo-2018].

Wikipedia. Googlebot — Wikipedia, the free encyclopedia. `http://support.google.com/webmasters/answer/182072`, 2017. [Online; fecha de acceso: 14-Mayo-2018].

Wikipedia. Bot conversacional — Wikipedia, the free encyclopedia. `http://es.wikipedia.org/wiki/Bot_conversacional`, 2018a. [Online; fecha de acceso: 14-Mayo-2018].

Wikipedia. Web scraping — Wikipedia, the free encyclopedia. `http://es.wikipedia.org/wiki/Web_scraping`, 2018b. [Online; fecha de acceso: 14-Mayo-2018].

wit.ai. Natural language for developers. `http://wit.ai`, 2018. [Online; fecha de acceso: 14-Mayo-2018].


This pro-
ject report was
typeset using the
LaTeX system origi-
nally developed by Les-
lie Lamport (1994), ba-
sed on TeX created
by Donald Knuth
(1986)